
Information-theoretic Graph Mining

Jing Feng



München 2015

Information-theoretic Graph Mining

Jing Feng

Dissertation

an der Fakultät für Mathematik, Informatik und Statistik

der Ludwig–Maximilians–Universität

München

vorgelegt von

Jing Feng

aus Xi'an, China

München, den 03 Dec 2014

Erstgutachter: Prof. Dr. Christian Böhm

Zweitgutachter: Prof. Anthony K. H. Tung, PhD

Tag der mündlichen Prüfung: 11 June 2015

Contents

Abstract	xvii
1 Introduction	1
1.1 Knowledge Discovery in Graph Data	2
1.1.1 Graph data	2
1.1.2 Graph Mining Tasks	3
1.2 Mining Knowledge from Multiple Types of Graphs	4
1.2.1 Simple Graph	6
1.2.2 Weighted Graph	6
1.2.3 Multi-relational Graph	7
1.2.4 Attributed Graph	7
1.2.5 Bipartite Graph	8
1.3 Information Theory for Graph Mining	9
1.4 Thesis Contributions	10
1.5 Thesis Organization	11
2 Theoretical Background	13
2.1 Basic Knowledge of Graph Theory	13
2.1.1 Graph Representation	13
2.1.2 Graph Topological Properties	15
2.1.3 Graph Clustering	16
2.1.4 Graph Structure Mining	23

2.1.5	Link Prediction	25
2.1.6	Graph Compression	27
2.2	Basic Knowledge of Information Theory	27
2.2.1	Information Theoretic Concepts	27
2.2.2	Minimum Description Length Principle	29
2.3	Evaluations in Graph Mining	31
3	Automatically Spotting Information-rich Nodes in Graphs	35
3.1	Introduction	36
3.1.1	Motivation	36
3.1.2	Contributions	37
3.2	The Information Content of A Node	39
3.2.1	Naive Coding Scheme	39
3.2.2	Information Content of A Node	40
3.3	Similarity Measure	40
3.4	Embedding	42
3.5	Algorithm Info-spot	43
3.5.1	MDL-based Coding Scheme	43
3.5.2	The Greedy Merging Algorithm Info-spot	45
3.5.3	Runtime Complexity	47
3.6	Experiments	47
3.6.1	Coauthor Network of 5 Famous Scientists	47
3.6.2	Data Mining Scientists Collaboration	51
3.6.3	Email Communication Network of 3 ex-CEO from Enron	51
3.7	Related Work and Discussion	53
3.8	Chapter Conclusion	54
4	Compression-based Graph Mining Exploiting Structure Primitives	57
4.1	Introduction	58

4.1.1	Motivation	58
4.1.2	Contributions	60
4.2	Graph Compression using Structure Primitives	61
4.2.1	Basic Coding Paradigm	62
4.2.2	Extended Coding Paradigm	66
4.3	Algorithm CXprime	68
4.3.1	Graph Partitioning	69
4.3.2	Link Prediction	70
4.4	Experimental Evaluation	72
4.4.1	Discovering the Graph Structure	72
4.4.2	Graph Partitioning	74
4.4.3	Link Prediction	81
4.5	Related Work and Discussion	83
4.5.1	Graph Structure and Pattern Mining	83
4.5.2	Compression-based Graph Mining	83
4.5.3	Graph Partitioning	84
4.5.4	Link prediction	84
4.6	Chapter Conclusion	85
5	Summarization-based Mining Bipartite Graphs	87
5.1	Introduction	88
5.1.1	Motivation	88
5.1.2	Contributions	89
5.2	Compressing a Bipartite Graph	90
5.2.1	Coding Scheme	92
5.2.2	Hidden Relations Between Vertices	94
5.3	Algorithm SCMiner	96
5.3.1	Basic Idea	97
5.3.2	Finding Super Node Candidates	97

5.3.3	Algorithm	99
5.3.4	Properties	99
5.4	Experiments	100
5.4.1	Data Sets	100
5.4.2	Clustering Quality	102
5.4.3	Hidden Structure	106
5.4.4	Link Prediction Accuracy	109
5.5	Related Work and Discussion	110
5.5.1	Co-clustering	110
5.5.2	Graph Compression and Summarization	111
5.5.3	Link Prediction	111
5.6	Chapter Conclusion	112
6	Detection of Overlapping Communities in Attributed Graphs	113
6.1	Introduction	114
6.1.1	Motivation	114
6.1.2	Contributions	116
6.2	Compressing an Attributed Graph	117
6.2.1	Notations	117
6.2.2	Coding Scheme	119
6.3	Algorithm IROC	124
6.3.1	Initialization	124
6.3.2	Refinement	125
6.3.3	Overall procedure	128
6.4	Experiments	129
6.4.1	Synthetic Data sets	130
6.4.2	Real Data sets	134
6.5	Related Work and Discussion	138
6.5.1	Attributed Graph Clustering	138

6.5.2	Detecting Overlapping Communities	140
6.6	Chapter Conclusion	141
7	Conclusions and Future Research	143
7.1	Spotting Information-rich Nodes in Graphs	144
7.2	Compression-based Graph Mining Exploiting Structure Primitives	144
7.3	Summarization-based Mining Bipartite Graphs	145
7.4	Finding Overlapping Communities in Attributed Graphs	146
	Acknowledgments	159

List of Figures

1.1	An Example of Collaboration Graph	5
2.1	Graph Representation	14
2.2	Examples of Typical Graph Clustering.	17
3.1	An Example of Using Regularities to Compress Graph and Detect Outstanding Nodes	38
3.2	Relation Pattern of Two Nodes	41
3.3	An Example of Graph	43
3.4	MDL Coding Example	44
3.5	The Process of Compression	48
3.6	Mapping Similarity Matrix of 5 Scientists Data Set to 4 Dimension and Plot in Each Two Pairs (Most interesting nodes are highlighted with their name).	50
4.1	Two Differently Structured Sub-graphs.	59
4.2	All Possible Connections of Three-node Primitives	63
4.3	Example for Coding and De-coding. (a) Graph; black edges: already coded; red edge: current edge; grey edges: not yet processed; (b) Adjacency matrix: filled entries: already coded in diagonal-wise order of processing; red: current entry; (c) Current stage of adjacency lists, see Section 4.2.1.	64
4.4	Multiple Primitives.	67
4.5	Coding Scheme Evaluation on Synthetic Data.	73

4.6	Coding Scheme Evaluation on Real-World Data.	74
4.7	Syn1 with Two Stars and One Clique.	76
4.8	Coding Cost of Clusters.	76
4.9	Syn2 with Three Stars.	77
4.10	Graph Partitioning of Zachary's karate club Data.	79
4.11	Precision of Synthetic Data Sets.	82
5.1	Tasks of SCMiner.	91
5.2	Summarization and Compression.	92
5.3	The Strategy Used for Merging Nodes	95
5.4	Results for Various ϵ	103
5.5	Cluster Purity on Movielens Data.	105
5.6	Hidden Structure Detected by SCMiner and CA. From Left to Right: Data Set BP1, BP2, BP3.	106
5.7	Hidden Structure of Cities Detected by CA (left) and SCMiner (right). For SCMiner results, Square C represents a cluster consisting of capitals, Square F financial cities, Square E economic cities and Square W is the isolated city Washington, DC. On the other side, the cluster represented by Circle C main- ly contains accountancy firms, Circle AB advertising and banking companies, Circle LB banking and law firms, and Circle L law firms.	107
5.8	Hidden Structure of MovieLens Detected by CA (left) and SCMiner (right). For SCMiner results, Square <i>O1</i> and <i>O2</i> denote clusters containing old users, Square <i>YW</i> represents a cluster of young women, and Square <i>YM1</i> and <i>YM2</i> young man. On the other side, the cluster represented by Circle <i>FM</i> represents high scored movies, Circle <i>CD</i> comedy and drama, Circle <i>CR</i> action, romance and comedy. Circles <i>AA</i> and <i>AS</i> represent adventure, action, thriller movies and action, sci-fi, thriller movies, respectively.	107
6.1	Motivational Example of a Social Friendship Network.	115
6.2	The Codebook of an Attribute Matrix	121

6.3	The Assignment List of Vertices	123
6.4	Clustering Results of Syn1 in Attributed Matrix.	133
6.5	Run Time of Varying Vertices.	133
6.6	Run Time of Varying Dimensionality.	134
6.7	Overlapping Between Cluster 2 and Cluster 11 of Ego-network “1912”.	136

List of Tables

3.1	Results of Info-spot and Oddball on 5 Famous Scientists Dataset	51
3.2	Results of Info-spot and Oddball on Data Mining Dataset	52
3.3	Results of Info-spot and Oddball on Email Communication Dataset	53
4.1	Distinguishing Graph Structure on Real Data Sets	74
4.2	Example of Differing People in MCL and CXprime	80
4.3	Compression Rates (Bits)	81
4.4	Precision of Real Data Sets (%)	83
5.1	Synthetic Bipartite Graphs	101
5.2	Clustering Performance on Synthetic Data.	102
5.3	Results on World Cities Data.	104
5.4	Link Prediction Performances.	109
6.1	Parameter Settings for Generating Synthetic Data Set	130
6.2	Evaluation Overlapping Clusters of Synthetic Data Sets	131
6.3	F-Measure of Facebook Data Sets	135
6.4	Overlapping of Ego-network “1912”	136
6.5	Subspace detected by IROC of Ego-network “1912”	137
6.6	F-Measure of Google+ Data Sets	138

Abstract

Real world data from various application domains can be modeled as a graph, e.g. social networks and biomedical networks like protein interaction networks or co-activation networks of brain regions. A graph is a powerful concept to model arbitrary (structural) relationships among objects. In recent years, the prevalence of social networks has made graph mining an important center of attention in the data mining field. There are many important tasks in graph mining, such as graph clustering, outlier detection, and link prediction. Many algorithms have been proposed in the literature to solve these tasks. However, normally these issues are solved separately, although they are closely related. Detecting and exploiting the relationship among them is a new challenge in graph mining. Moreover, with data explosion, more information has already been integrated into graph structure. For example, bipartite graphs contain two types of node and graphs with node attributes offer additional non-structural information. Therefore, more challenges arise from the increasing graph complexity. This thesis aims to solve these challenges in order to gain new knowledge from graph data.

An important paradigm of data mining used in this thesis is the principle of Minimum Description Length (MDL). It follows the assumption: the more knowledge we have learned from the data, the better we are able to compress the data. The MDL principle balances the complexity of the selected model and the goodness of fit between model and data. Thus, it naturally avoids over-fitting.

This thesis proposes several algorithms based on the MDL principle to acquire knowledge from various types of graphs: Info-spot (Automatically Spotting Information-rich Nodes in Graphs) proposes a parameter-free and efficient algorithm for the fully automatic detection of

interesting nodes which is a novel outlier notion in graph. Then in contrast to traditional graph mining approaches that focus on discovering dense subgraphs, a novel graph mining technique CXprime (Compression-based eXploiting Primitives) is proposed. It models the transitivity and the hubness of a graph using structure primitives (all possible three-node substructures). Under the coding scheme of CXprime, clusters with structural information can be discovered, dominating substructures of a graph can be distinguished, and a new link prediction score based on substructures is proposed. The next algorithm SCMiner (Summarization-Compression Miner) integrates tasks such as graph summarization, graph clustering, link prediction, and the discovery of the hidden structure of a bipartite graph on the basis of data compression. Finally, a method for non-redundant graph clustering called IROC (Information-theoretic non-Redundant Overlapping Clustering) is proposed to smartly combine structural information with non-structural information based on MDL. IROC is able to detect overlapping communities within subspaces of the attributes.

To sum up, algorithms to unify different learning tasks for various types of graphs are proposed. Additionally, these algorithms are based on the MDL principle, which facilitates the unification of different graph learning tasks, the integration of different graph types, and the automatic selection of input parameters that are otherwise difficult to estimate.

Zusammenfassung

Reale Daten von unterschiedlichsten Domänen können als Graphen modelliert werden, z.B. soziale Netzwerke und biomedizinische Netzwerke wie Protein-Interaktions Netzwerke oder Co-Aktivierungsnetzwerke von Gehirnregionen. Ein Graph ist ein mächtiges Konzept, um beliebige (strukturelle) Beziehungen zwischen Objekten zu modellieren. In den letzten Jahren ist gerade durch die Verbreitung sozialer Netzwerke das Graph Mining zu einem wichtigen Fokus innerhalb des Data Mining Forschungsgebiets geworden. Es existieren viele wichtige Aufgaben innerhalb des Graph Minings, wie z.B. Graph Clustering (Gruppierung der Knoten), Outlier Detection (Ermittlung von Ausreißern) und Link Prediction (Vorhersage von Kanten). Viele Algorithmen zur Lösung dieser Aufgaben wurden entwickelt. Allerdings werden diese Probleme oft einzeln gelöst, obwohl sie eine enge Beziehung zueinander haben. Diese Beziehung aufzuzeigen und zu nutzen ist eine neue Herausforderung im Graph Mining. Außerdem werden mit der zunehmenden Datenexplosion immer mehr Informationen in die Graphstrukturen integriert. Zum Beispiel beinhalten bipartite Graphen zwei Typen von Knoten. Graphen mit Attributen an den Knoten bieten zusätzliche nicht-strukturelle Informationen. Daher treten durch diese erhöhte Komplexität immer weitere Herausforderungen auf. Diese Doktorarbeit versucht diese Herausforderungen zu lösen um neues Wissen von Graphdaten zu erhalten.

Ein wichtiges Data Mining Paradigma, das dieser Doktorarbeit zugrunde liegt, ist das Prinzip der “Minimum Description Length” (MDL, minimale Beschreibungslänge). Es folgt der Annahme, dass, je mehr Wissen wir von den Daten erfahren können, desto besser können wir die Daten komprimieren. Das MDL Prinzip balanciert die Komplexität eines gewählten Modells mit der Güte wie dieses Modell auf die Daten passt. Damit vermeidet es automatisch, dass

das Modell zu komplex wird (so genanntes over-fitting). Diese Doktorarbeit stellt mehrere Algorithmen vor, die auf dem MDL Prinzip beruhen um Wissen über diverse Graphtypen zu erhalten: Info-spot stellt einen Parameter-freien und effizienten Algorithmus vor um vollautomatisch die interessantesten Knoten zu erkennen. Dies stellt eine neue Definition des Konzepts “Outlier” in Graphen dar. Danach stellen wir im Gegensatz zu traditionellen Graph Mining Ansätzen, die sich auf die Entdeckung dichter Subgraphen fokussieren, eine neue Graph Mining Technik namens CXprime vor. Sie modelliert die Transitivität und die Anzahl von Hubs in einem Graphen durch Strukturprimitive (alle Kombinationen von Kanten in drei Knoten). Das Kodierungsschema von CXprime ist in der Lage, Gruppen mit Strukturinformationen ausfindig zu machen, dominierende Unterstrukturen eines Graphen zu unterscheiden und Kanten unter zu Hilfe nahme dieser Unterstrukturen vorherzusagen. Der nächste Algorithmus SCMiner integriert Aufgaben wie Graphsummierung, Graph Clustering, Kantenvorhersage und die Entdeckung von versteckten Strukturen innerhalb eines bipartiten Graphen auf der Basis von Datenkompression. Zuletzt ist eine Methode namens IROC für nicht-redundantes Graphclustering gegeben um sinnvoll Strukturinformation mit nicht-struktureller Information basierend auf MDL zu kombinieren. IROC ist dazu in der Lage überlappende strukturelle Gruppierungen innerhalb der Unterräume der Attribute eines Graphen zu finden.

Zusammengefasst werden Algorithmen vorgestellt um verschiedene Lernaufgaben mit unterschiedlichen Graphtypen zu vereinen. Zusätzlich basieren alle diese Algorithmen auf dem MDL Prinzip, welches die Vereinigung von verschiedenen Graphlertechniken, die Integration von unterschiedlichen Graphtypen und die automatische Selektierung von Eingabeparametern, die sonst schwierig zu schätzen wären, überhaupt erst ermöglicht.

Chapter 1

Introduction

With the rapid development of data generation and collection techniques, people have gradually realized the importance of data. The explosive growth rate stimulates people's desires to make profit from data. Thus the technique called "Data Mining" that automatically extracts useful knowledge from data has emerged during the last decades. It aids people to understand the current situation or make decisions for the future by discovering the knowledge behind data. For decades, the technique is diversified by developing various methods for mining data stored in different types, such as numerical data, categorical data, relational data and etc.

In the new century, when human beings step into the Internet era, the quantities and complexities of the obtained data have dramatically increased. Due to the powerful ability of modeling the relationship of the complex data, graph structure is gradually attracting more and more attentions. In real life, graphs can be used to model many interesting events from various fields, including social networks, web connections, road maps, protein-protein interaction networks and etc. Therefore, in recent years graphs become one of the most popular structures that data scientists used to analyze data, and graph mining that aims to discover knowledge from graph data has become one of the most important branches of data mining.

In this chapter, first and foremost, Section 1.1 briefly introduces the concept of knowledge discover in graph data. Then Section 1.2 further elaborates mining knowledge from different types of graph. In Section 1.3, information theory for graph mining is introduced. Section

1.4 concludes the proposed algorithms and the contributions. Finally, Section 1.5 displays the structure of the thesis.

1.1 Knowledge Discovery in Graph Data

A graph consists of a set of nodes and links between them. Normally, they are named as vertices and edges respectively. Knowledge discovery in graph data is called graph mining which aims to extract useful and novel knowledge from graph data. In this section, we firstly describe the merits of graph structure and then introduce some graph mining tasks.

1.1.1 Graph data

Why graph data is becoming more prevalent? Firstly, in mathematic and computer science, graph is the data structure that lays emphasis on depicting the relationships among objects. With the increasing of data complexity, people gradually concern more about modeling data objects by their relationships. With vertices representing data objects and edges expressing relationships between pairs of data objects, graph is a general data structure that can represents any kind of data. Graph holds the merit of well expressing the structure of the data. Moreover, data in many application fields, like social network, biology, commerce, can be modeled as graph. Specifically, the most popular and successful applications of graph modeling is in social network, like Facebook and Twitter, where users are considered as vertices and their relationships are regarded as edges. Graph is used in biology field as well, where vertices represent genes or proteins and edges represent their interactions. Furthermore, graph is also used in modeling Internet. Here, vertices represent the web pages and edges are hyper-links in between. In brief, modeling these real data as graph reveals the interaction among data objects, which has the potential to aid us further understanding the data. Due to the fact that graph is a powerful data modeling tool and it is widely applicable in real life, this thesis mainly focuses on graph mining.

1.1.2 Graph Mining Tasks

In general, there are many ways to mine novel and useful knowledge from graph data, which can be categorized into different graph mining tasks, e.g. anomaly detection, community detection, link prediction, etc. In the following, we elaborate those ones that are studied in this thesis in more details.

Firstly, finding out special outstanding vertices from a large graph is a very interesting task. Vertices in a graph may play different roles, due to the diverse connections. Specifically, some vertices connecting with many other vertices are considered as hubs or centers of the graph. Some vertices connecting two or more densely connected clusters play as the role of bridges. Some vertices form unique structure with their neighbors, e.g. clique. One example in real data is identifying which person is the most active user in a group of Facebook or discovering which gene is the key factor to form an important structure in biology data.

Secondly, analyzing the topological structure contributes to understand the overall trends of data. Different connections lead to various topological structures of graphs. For example, social network data usually exhibits small-world phenomenon which means that connections between pairs of vertices are relatively tight. Web data usually shows itself as the combination of many spoke structures because main pages always link to many sub-pages.

Thirdly, link prediction is a practical task for many applications. Based on the existing edges, we can predict the missing links or the link which will appear in the future. It helps social networking site to recommend friends or applications to users, and it also aids biologist to discover new connections between genes or proteins.

Moreover, discovering special structures from graph data is an effective way to understand it. Vertices and edges can form special structures, like tree structures, densely connected clusters, star structures and etc. Graph clustering or community detection is one important task that arranges nodes with common characteristics into groups. To be specific, nodes in clusters may contain similar characters, possess similar connections or have tight relationships. For example, by graph clustering we can judge which authors are in the same research field in collaboration network, and we can analyze which proteins provide similar functions in protein interaction net-

work.

Next, substructure mining discovers small structures that appear frequently in a large graph. The large graph can be simplified by these frequently appearing substructures. Thus in social network we can analyze how people construct their friend circles. Besides, we can find that which kinds of combined genes often appear in the protein. Additionally, some nodes form unique structures in the graph. For example, nodes grouping as a clique depicts tight relationships, nodes forming as a star expresses a central structure and nodes taking on a tree structure presents a hierarchical structure.

Overall, the tasks for mining graph data are not limited to the above. Graph data is complex, thus there are still much more interesting knowledge need to be discovered. Moreover, each graph mining task is not independent. They are closely interrelated. For example, graph clusters are also graph substructures. Links have higher probabilities to exist among the vertices in the same cluster. Frequent appearing substructures can be abstracted to summarize a graph. It is interesting to adopt the interrelationship of these graph mining tasks and to propose algorithms that are able to achieve multiple tasks at the same time.

1.2 Mining Knowledge from Multiple Types of Graphs

As the complexity of real data is increasing, the obtained graph does not only consist of normal vertices and edges. Moreover, vertices and edges may associate with some additional information, which leads to variety of graph types. Figure 1.1 shows a group of graphs with different types which are generated from the collaboration field. Vertices are authors and papers, edges are relations between them. As shown in the figure, graph type can be diversified by adding extra edge information or vertex information. In the following, we will introduce these types of graph and the challenges of mining these graphs in more details.

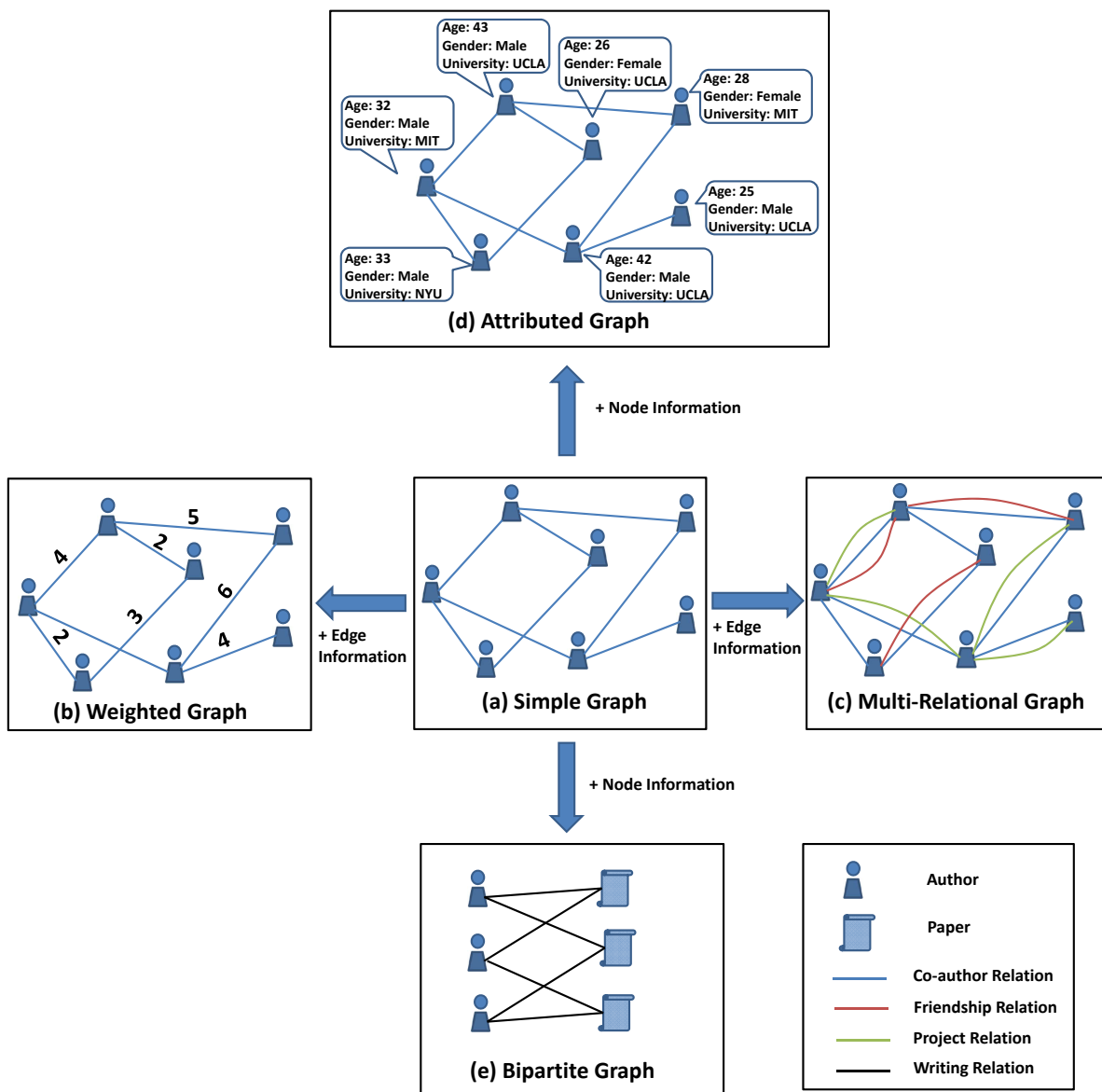


Figure 1.1: An Example of Collaboration Graph

1.2.1 Simple Graph

A set of objects with pairwise relationships can be naturally modeled as a graph. Objects are represented as vertices, and any two objects that have some certain relationship are joined by an edge. The existence of the directions of the edges distinguishes directed graph from undirected graph. In this thesis, we only consider the undirected situation without considering the orientation of the edges. An example of a simple undirected graph is shown in Figure 1.1 (a). In the example graph, vertices stand for authors and edges represent co-author relationships. The whole graph represents the cooperative relationship of a group of authors. In general, the graph clearly expresses the ability of representing the relationships of all the objects by the links.

Mining simple graphs has been concerned for a while. Many algorithms have been proposed to fulfill different tasks described in Section 1.1.2, such as graph clustering, link prediction, substructure mining, etc. Challenges of mining such simple graphs are to improve the efficiency and effectiveness of these methods as well as to discover novel knowledge that classical methods cannot find out. For example, comparing with the classical algorithms, new algorithms will be able to improve the quality of the clustering results. In other cases, they are able to discover some novel knowledge, like the new type of clusters or structures.

1.2.2 Weighted Graph

A graph is named as weighted graph if its edge associates with a numerical value. The numerical value is called weight which normally expresses the strength of the connection. Usually, the weight is a non-negative value. Figure 1.1 (b) is a weighted graph example coming from a collaboration network. In the example graph, the edges express the cooperative relationship and the weight expresses the number of papers the two authors have published together. Therefore, weighted graph can be transferred from a simple graph by considering the strengths of the links.

For mining the weighted graph, we need to consider not only the connections but also their strengths. To be specific, when clustering the weighted graph, we need to add weight to the measurement of density of the connection. Substructures contain higher weights are considered more interesting. In link prediction, not only future links need to be predicted but also the

strength of the links.

1.2.3 Multi-relational Graph

In real life example, there may be more than one relationship between two objects. For instance, two people can be not only colleagues but also friends. Thus we introduce multi-relational graph which may contain more than one type of edges between same pair of nodes. Figure 1.1 (c) is an example of a multi-relational graph from the collaboration field. Edges with different colors stand for different relationships: blue edges represent the co-author relationship, red edges indicate friendship relations and green edges express colleague relations. Therefore, comparing with simple graph, multi-relational graph contains more link information, and consists of more complex structures.

Mining multi-relational graph needs to consider the relations between multiple types of edges. The mining process on such graph is influenced by all the types of edges. Intuitively, different types of community can be discovered from it. Regarding link prediction, with multi-relational graph we can not only predict the edges, but also specify the type of edge which can be the explanation of the predication. It is a challenge to find out these information from the multi-relational graph.

1.2.4 Attributed Graph

Vertices with attributes provide additional information to graphs. Therefore, an attributed graph is defined as a simple graph plus additional node information. The additional node information can be a numerical or categorical vector which indicates the characteristics of the vertices. For example, Figure 1.1 (d) is an attributed graph which is generated from the simple graph Figure 1.1 (a) by considering the extra node information. Similarly, the graph demonstrates the cooperation relationship of the authors. Simultaneously, we consider the author's information, like age, gender, university and etc.

The importance of mining attributed graph is the balance of the structure information and the feature information. Take the graph clustering as an example, from the point view of structure,

clusters should be densely connected, while from the point view of attributes, clusters should contain similar attributes. Finding the best result by considering both influence factors is a challenge.

1.2.5 Bipartite Graph

Figure 1.1 (e) is an example of a bipartite graph which contains two types of vertices and the edges only exist between different type of vertices. The example bipartite graph shows the relationship between authors and papers. If an author writes a paper, there is an edge between them. A bipartite graph is transferred from a simple graph by adding a new type of vertex. Thus it can be projected to a simple graph. Take Figure 1.1 (e) as an example, the co-authorship is hidden in the bipartite graph. If two authors both write a paper, there is a cooperation relation between them. Thus it can be projected to a simple graph with edges representing co-authorship. Moreover, with the increasing of the number of vertices type, tri-partite or k-partite graph can be also generated. The character of such k-partite graph is that edges only exist between different types of vertices.

For such types of graph, finding out the relations of multiple types of vertices is very interesting. Traditionally, clusters exist only on each type of vertices. Thus the relationship among these clusters need to be explored. Due to the increasing diversity of obtained data, graph is not confined to only one type of additional information. All the additional information can exist simultaneously. For example, weighted attributed graph, weighted bipartite graph or even more complex graph. Combine all these information to mine novel knowledge from such graph is another challenge.

Overall, as the Figure 1.1 shows, data from the same field can be modeled as different types of graph when adopting different information. Obviously, the more additional information is contained, the more complex the graph data is. Mining the graph becomes more and more difficult as the complexity of the graph increases. However, each type of graph has their own characteristic. The knowledge discovered from them has different emphasis. Therefore, there are many challenges to explore different types of graph as mentioned before. This thesis focuses

on mining multiple types of graphs and tries to solve most of these challenges.

1.3 Information Theory for Graph Mining

Various algorithms are proposed to solve the problem of graph mining. Many of them suffer from parameterization issue. For example they need to set the number of clusters or they need to predetermine several thresholds. Usually the algorithms need to run several times to obtain the best parameters. In addition, without domain expert it is hard to determine which result is better. Information theory is one of the techniques that can be used to avoid this parameterization problem. Many information criterion are proposed for model selection, which is also applicable for graph mining. These information theoretic methods own the potential to explore patterns (clusters, substructures and etc.) from the graph data without parameters.

Information theory and data mining are both branches of information science. Data mining focuses on discovering useful information from massive and complex data, while information theory lays emphasis on describing and expressing the data by quantification of its information. The purpose of data mining is to mine useful knowledge from the data. However, what kind of knowledge is useful? Some information theoretic concepts (like entropy, mutual information and etc.) can be used to measure the usefulness. In Chapter 2 we will introduce them in more details. On the other aspect, lossless data compression is one important application of information theory. It compresses the data by the regularity inside data, which can be used in data mining to discover the rules and regularity as well. Minimum Description Length (MDL) principle [92] is an information criteria that is based on lossless compression. It measures all type of information by the length of binary string which can be obtained by encoding the information. We adopt MDL as the core technique of the thesis, due to its ability to discovery the regularity inside data. We will introduce more details of the technique in Chapter 2.

Several advantages enable us to adopt information theory to solve graph mining problem. To be specific, information theoretic concepts can be used to measure similarity of link patterns and substructures. Moreover, as graph data can be represented by an adjacency matrix containing only '0's and '1's, patterns or rules can be obtained from the graph by compressing the adjacency

matrix. With the increasing of the complexity of graph data, more regularities can be found by compression, thereby obtaining more information from the data. Furthermore, Minimum Description Length principle is able to select the best model. Once many patterns have been detected, MDL can tell which ones are more informative. Overall, by adopting information theory concept, we propose several algorithms to mine useful knowledge from multiple types of graph data in this dissertation.

1.4 Thesis Contributions

The central idea of the thesis is mining knowledge from graph data by adopting information theoretic technique, especially the Minimum Description Length principle (MDL). Firstly, the thesis aims at proposing algorithms to fulfill multiple graph mining tasks. For instance, the algorithm is able to achieving graph clustering, link prediction, substructure mining at the same time. Secondly, the thesis aims at mining knowledge from different types of graphs. In the thesis, we focus on mining simple graph, bipartite graph and attributed graph. Moreover, based on MDL principle all the proposed algorithms are parameter-free. For different graph type, various coding schemes are proposed to achieve different aims. The following are contributions of the proposed algorithms.

1. A novel algorithm **Info-spot** is proposed to automatically detect interesting nodes in large graphs. In Info-spot, interesting nodes are considered as a novel outlier notion which is defined by compressing the link patterns of the nodes. And an intuitive similarity measure is proposed to measure the difference between pair of link patterns. Guided by the MDL principle, the algorithm can detect the interesting nodes automatically. Parts of the material presented in this chapter have been published in [47].
2. A novel compression-based graph mining algorithm **CXprime** is proposed based on exploiting three-node primitives which are the smallest substructures that can express both transitivity and hubness of a graph. A novel coding scheme based on three-node primitives is proposed. Guided by the MDL principle, frequently appearing primitives are effectively

compressed. And the proposed algorithm is able to distinguish graphs, to partition graphs and to predict missing links based on different substructures. Parts of the material presented in this chapter have been published in [30].

3. A technique **SCMiner** integrating summarization, clustering and link prediction on bipartite graphs is proposed. The basic idea is to transform the original graph into a very compact summary graph which is controlled by the MDL principle. The proposed algorithm discovers the major clusters of both vertex types as well as the major connection patterns between those clusters and even predict links between different type of vertex. Parts of the material presented in this chapter have been published in [32].
4. A new method **IROC** is proposed to clustering attributed graphs with the objective to find the reasonable overlapping communities and meaningful subspace of the attributes at the same time based on an information theoretic technique. Parts of the material presented in this chapter have been submitted in [31].

1.5 Thesis Organization

The remainders of the thesis are organized as follows: Chapter 2 further introduces the theoretical background of the thesis. We describe basic knowledge of graph mining including graph representation, graph properties and research status of graph mining tasks. We also briefly describe basic knowledge of information theory, especially the Minimum Description Length principle (MDL). Chapter 3 addresses interesting nodes detection problem. We introduce a MDL-based algorithm aiming to detect interesting nodes from general large graph. Chapter 4 concentrates on mining graph based on graph structure. We introduce a novel coding scheme based on compressing three-node primitives. And then clusters with different cluster type based on structures can be detected. Chapter 5 develops a integrating algorithm of bipartite graph which combine graph compressing, graph clustering and link prediction. The chapter introduces a bipartite mining algorithm based on MDL aiming to find clusters, hidden structures and missing links. Chapter 6 concentrates on clustering problems of graph with attributes. The chapter proposes a algorithm

based on information theory to detect overlapping clusters together with the interesting subspace of attributes. Finally, Chapter 7 concludes the thesis and states the further research directions.

Chapter 2

Theoretical Background

In this chapter, we introduce the background knowledge of both graph mining and information theory in more details. First of all, we describe the basic knowledge of graph including graph representation and some graph topological properties. Then we survey the existing algorithms for the classical problems of graph mining, e.g. graph clustering, substructure mining, link prediction and graph compression. In section 2.2, we further introduce the background knowledge of information theory. We start with several information theoretic concepts [122], such as entropy and mutual information. Then the lossless compressing technique Minimum Description Length principle (MDL) is introduced. In the end of this chapter, we list some evaluation methods which are adopted in the thesis for each graph mining tasks.

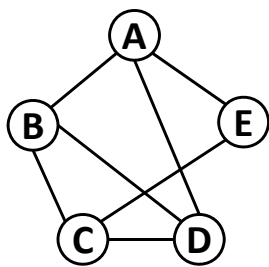
2.1 Basic Knowledge of Graph Theory

2.1.1 Graph Representation

Numerous of problems can be solved by using graph, due to its advantages of providing a different point of view and making the problems much simpler. It is important to understand different types of representation of graph data. In this part, we mainly consider the undirected graph. First of all, the definition of graph is give as follow.

Definition 2.1 (Graph) A simple graph is defined as $G = (V, E)$, with V representing a set of vertices and E standing for all the edges which connect pairs of vertices.

Graph Visualization. When there are not so many vertices or edges, visualization is an effective way to represent the graph data, due to the clear displaying of the edge connections. For example, Figure 2.1 (a) is a visualization of a simple graph. Intuitively, graph is visualized as the combination of points and lines. The points represent the vertices and the lines represent the edges. From the graph visualization, we can obtain which vertices are isolate, which vertices are densely connected and which vertices form star structure or triangular structure. For example, in Figure 2.1 (a), it is clearly shown that vertex B, C, D form a triangle structure. However, the scale of the contemporary data is very large. Thus the visualization of big graph is mass and it is not possible to obtain any useful information from it. Visualization is not appropriate to represent graphs with thousands or even hundreds of vertices.



(a) Graph

	A	B	C	D	E
A	0	1	0	1	1
B	1	0	1	1	0
C	0	1	0	1	1
D	1	1	1	0	0
E	1	0	1	0	0

(b) Adjacency Matrix

Vertices	Neighbors
A	B,D,E
B	A,C,D
C	B,D,E
D	A,B,C
E	A,C

(c) Adjacency List

Figure 2.1: Graph Representation

Adjacency Matrix. In mathematics, adjacency matrix is a typical way to represent a graph. For an undirected simple graph, the adjacency matrix is a symmetric binary matrix, as shown in Figure 2.1 (b). The rows and columns represent all the vertices respectively and entries indicate relations between pairs of vertices. To be specific, the entry displayed as ‘1’ means there is an edge between the corresponding two vertices and ‘0’ means the two vertices do not have connection. Therefore, adjacency matrix contains the connection information of each vertex.

Moreover, various types of graph data are represented as different types of adjacency matrices. To be specific, for weighted graph, entries of the adjacency matrix are written as weights instead of '1's. Multi-relational graph needs multiple adjacency matrices to represent all the relations. The row and the column of bipartite graph represent the different type of vertices separately. And the number of rows and columns is not equal. Attributed graph need an extra matrix to represent the attributes of the vertices.

Adjacency List. Additionally, adjacency list is another way to represent the graph. As shown in the Figure 2.1 (c), adjacency list is an list with the length equals to the number of vertices. It gives all the neighbors of each vertex. Comparing with the adjacency matrix, the adjacency list is more suitable to sparse graph and is more space-saving.

2.1.2 Graph Topological Properties

The amount of vertices and edges decides the size and the density of the graph respectively. And different connections lead to variations in graph topological structures. In the following, we will introduce some basic global and local properties of the graph data which are common in mining the graph data.

Connectivity. Connectivity is the basic characteristic of graph. Based on this characteristic, one can get from one node to any other node by following a sequence of edges.

Degree. Degree is the concept with regard to the vertex. The degree of a vertex v is defined as the number of edges connecting to the vertex v . Degree to a certain extent reflects the importance of the vertex. The vertex with a large degree denotes that the vertex has relations with many other vertices. And the vertex with 0 degree is an isolate vertex.

Path. Path is a edge sequence which is produced by traversing from one vertex to another vertex. There are many possible paths between two vertices. The most important one is the shortest path which is defined as the path between two vertices with the smallest length. The shortest path between two vertices is unique. In general graph, the shortest path is the path with the smallest number of edges, while in weighted graph the shortest path is the path with the smallest sum of the weights. And the shortest path can be used to measure the distance between the two vertices.

Clustering Coefficient. Watts and Strogatz [117] mention and popularize the concept “clustering coefficient”. For each vertex, we calculate the local clustering coefficient which measures the degree of a vertex that can form cluster with other vertices. It is calculated from Eq. (2.1), where $N_{triangles}(v_i)$ is the number of triangles formed by vertex v_i and $N_{triples}(v_i)$ is the number of triples formed by vertex v . For the whole graph, we calculate the global clustering coefficient, as shown in Eq. (2.2), which equals to the average of local clustering coefficient of all the vertices, where $|V|$ is the number of vertices.

$$C(v_i) = \frac{N_{triangles}(v_i)}{N_{triples}(v_i)} \quad (2.1)$$

$$C = \frac{1}{|V|} \sum_{i=1}^{|V|} C(v_i) \quad (2.2)$$

Betweenness Centrality. Betweenness centrality [34] is an index measuring the importance of the vertex that acts as a bridge in the graph. Betweenness centrality of a vertex is the ratio of the number of the shortest paths from vertex v_m to vertex v_n through vertex v to the number of all shortest paths from v_m to v_n which is shown in Eq. (2.3).

$$Betweenness(v) = \sum_{v_m \neq v \neq v_n} \frac{Path_{v_m v_n}(v)}{Path_{v_m v_n}}. \quad (2.3)$$

2.1.3 Graph Clustering

Clustering is a typical technique for analyzing the data set, which aims to find groups of data sharing similar concepts. In numerical data, distance-based clustering algorithm like K-means [73] aims to find groups of data with smaller distances in the groups and the larger distances among the groups. Density-based clustering algorithm like DBSCAN [29] aims to detect groups of data which contain higher density in the group while lower density between the groups. In graph data, what do graph clusters look like? Normally, as shown in Figure 2.2, the substructure with vertices densely connected is considered as a graph cluster. In the following, we will introduce several types of graph clustering which is related with the proposed algorithms.

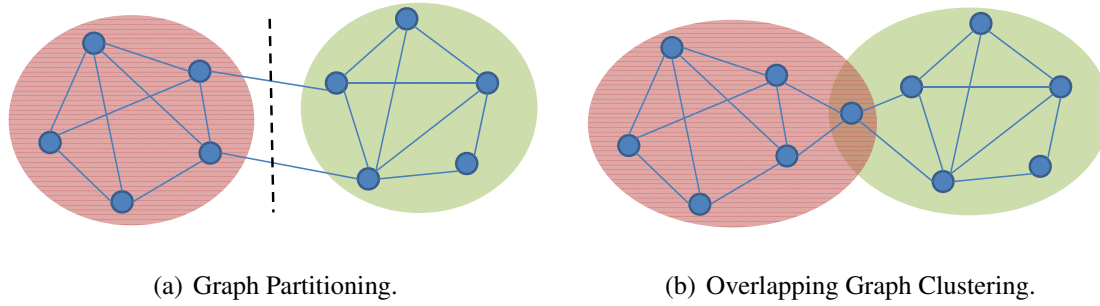


Figure 2.2: Examples of Typical Graph Clustering.

Graph partitioning

Graph partitioning, defined in Definition 2.2, is a technique that divides the vertices into several components. And each vertex can only be assigned to one component.

Definition 2.2 (Graph Partitioning) *Given a graph $G = (V, E)$ with V representing all the vertices and E standing for all the edges, graph partitioning divides vertices V into K components $\{C_1, C_2, \dots, C_K\}$ and fulfills the conditions that $C_1 \cup C_2 \cup \dots \cup C_K = V$ and $\forall i \neq j, C_i \cap C_j = \emptyset, i = \{1, 2, \dots, K\}, j = \{1, 2, \dots, K\}$.*

Normally, edges in each component are densely connected, while edges between each pair of components are sparsely linked. This is similar with the clustering in numerical data. Thus it is named as graph clustering. Figure 2.2(a) is a simple example of graph partitioning. The graph is divided into two densely connected components by cutting the sparse edges. Therefore, graph cuts are intuitive approaches for partitioning the graph. Normally, graph cuts are designed for bisection which remove the edges between the two parts. In graph $G = (V, E)$, graph cut $Cut(C_1, C_2)$ is defined as Eq. (2.4), where C_1 and C_2 are two graph partitioning components of vertices V , v_i and v_j are vertices belonging to subset C_1 and C_2 separately, $w(v_i, v_j)$ is the weight between vertex v_i and v_j . Thus cut is the number of edges between two disjoint subsets in the unweighted graph and is the sum of the weights of edges between two disjoint subsets in the weighted graph. Many graph cuts are proposed to solve the graph partition problem. In the

following, we take two famous graph cuts: Minimum cut and Normalized cut as examples.

$$Cut(C_1, C_2) = \sum_{v_i \in C_1, v_j \in C_2} w(v_i, v_j). \quad (2.4)$$

Minimum cut [44], shown in Eq. (2.5), needs the number of edges or the sum of weights between the component C_1 and C_2 to be minimum. Then the cut is extended to k-way cut to fulfill the demand of multiple clusters. Minimum k-cut [43] is defined in Eq. (2.6). The cut needs the sum of the number of edges or the sum of weights between every two components to be minimum.

$$MinCut(C_1, C_2) = \min \sum_{v_i \in C_1, v_j \in C_2} w(v_i, v_j). \quad (2.5)$$

$$KMinCut(C_1, C_2, \dots, C_K) = \min \sum_{i=1}^{K-1} \sum_{j=i+1}^K \sum_{v_m \in C_i, v_n \in C_j} w(v_m, v_n). \quad (2.6)$$

Normalized cut [99] of two parts is defined as Eq. (2.7), where $assoc(C_1, V)$, shown in Eq. (2.9), is the sum of weights between vertices in subset C_1 and all the vertices in V , and $assoc(C_2, V)$ is defined similarly. Comparing with minimum cut, normalized cut has the merit of avoiding over small size of partitions. And it can be easily extended to a k-way normalized cut as shown in Eq. (2.8).

$$NCut(C_1, C_2) = \frac{Cut(C_1, C_2)}{assoc(C_1, V)} + \frac{Cut(C_1, C_2)}{assoc(C_2, V)}. \quad (2.7)$$

$$KNCut(C_1, C_2, \dots, C_K) = \sum_{i=1}^K \frac{Cut(C_i, V - C_i)}{assoc(C_i, V)}. \quad (2.8)$$

$$assoc(C_1, V) = \sum_{v_i \in C_1, v_j \in V} w(v_i, v_j). \quad (2.9)$$

Moreover, Newman proposed the concept of modularity [82], which measures the ability of the network divided into communities also named as clusters. The object function which shown in Eq. (2.10), is generated based on the idea that the fraction of the edges in the clusters should

be larger than the expected fraction in the random graph, where $m = \sum_{i=1}^N \text{deg}(v_i)$ is the total number of edges, N is the number of vertices of the graph, A is the adjacency matrix of the graph with $A_{v_i, v_j} = 1$ standing for the connection between v_i and v_j , $\text{deg}(v_i)$ is the degree of vertex v_i , and $\delta(C_{v_i}, C_{v_j}) = 1$ if v_i and v_j belong to the same cluster or $\delta(C_{v_i}, C_{v_j}) = 0$. Higher Q indicates that dense connection within clusters and sparse connection between clusters. In some network paper, graph clustering is also named as community detection.

$$Q = \frac{1}{4m} \sum_{i=1}^N \sum_{j=1}^N \left(A_{v_i, v_j} - \frac{\text{deg}(v_i)\text{deg}(v_j)}{2m} \right) \delta(C_{v_i}, C_{v_j}). \quad (2.10)$$

These graph cuts and modularity based methods detect clusters based on the structures. The optimization of the graph cuts and modularity is NP-hard. Many methods are adopted to solve this problem to achieve local optimal. For example, graph cuts are normally combined with spectral partitioning methods [26, 24] using eigenvectors of the adjacency matrix to obtain the optimal partition of the graph and the simulated annealing strategy is used to maximize the modularity of a network [69]. Initially, these cuts are proposed for bisection. Although they are able to be extended to detect multiple clusters, the number of clusters needs to be predefined.

Many classical graph partition algorithms are proposed during last decade. In the following, we briefly introduce METIS and MCL which are adopted as comparison methods of our proposed algorithm CXprime [30] in Chapter 4. METIS [56] is a framework containing multilevel algorithms for graph partitioning. Multilevel graph partitioning algorithms [55, 49] mainly contain three procedures: coarsening phase, partitioning phase and uncoarsening phase. During the coarsening phase, graph G is transformed into a sequence of small graphs G_1, G_2, \dots, G_m . Each small graph contains fewer vertices, $|V_G| > |V_{G_1}| > \dots > |V_{G_m}|$. In this phase, several methods can be used to achieve these small graphs. For example, they find a random matching and collapse the matched vertices to form the vertex of the next level graph [14, 6] or form the vertex of the next level graph by highly connected group of vertices [18]. Then the obtained small coarse graphs are partitioned. Spectral bisection [88, 6], geometric bisection [76] and combinatorial methods [35] are adopted to achieve this procedure. In the uncoarsening phase, the partition results are projected to the original graph to obtain the results. Kernighan-Lin (KL) partition [33]

is another classical method which is able to obtain good results. Compared with spectral graph partitioning, multilevel graph partitioning possesses high efficiency, but it still needs to set the number of cluster in advance.

Markov Clustering, abbreviated as MCL [112], is based on the idea that a random walk starting from one node has a high probability to reach the next node in the same dense area. By iteratively implementing expansion and inflation on transition probability matrix of the graph, the MCL algorithm aims to encourage the random walk staying in the same dense area and punish the random walk between different dense area. Thus until the transition probability matrix convergence, graph clusters can be obtained. MCL do not need the number of clusters as the input parameters, but it needs to set the inflation parameter which affects the number of cluster.

Overlapping Graph Clustering

In many real life graphs, connections among vertices are complicated. Some vertices have tight connections with many clusters, thus assigning such vertices to one cluster as described in graph partitioning will loss some link information. Therefore, we can detect overlapping graph clusters instead of graph partitioning. Overlapping graph clustering is defined in Definition 2.3, which is a technique that divides the vertices into several components allowing overlaps among clusters. It means that vertices can be assigned to multiple clusters. In some social network examples, people have multiple identities in the network, like friends, colleagues and classmates. This is an intuitive example of overlapping phenomenon in the graph. Figure 2.2(b) shows an example of overlapping graph clustering, in which two dense clusters with overlapping vertex are detected.

Definition 2.3 (Overlapping Graph Clustering) *Given a graph $G = (V, E)$ with V representing all the vertices and E standing for all the edges, overlapping graph clustering detects K dense components $\{C_1, C_2, \dots, C_K\}$ of vertex V and fulfills the conditions that $C_1 \cup C_2 \cup \dots \cup C_K = V$ and $\forall i \neq j, \exists C_i \cap C_j \neq \emptyset, i = \{1, 2, \dots, K\}, j = \{1, 2, \dots, K\}$.*

In recent years, many algorithms are proposed to detect overlapping clusters in the graph. Palla and Imre [86] first reveal overlapping phenomena of complex networks in nature. They achieve overlapping communities by seeking k-cliques which contain overlapping vertices. The survey

paper [118] mentions numerous algorithms for detecting overlapping communities. To mention a few, Gregory proposes several algorithms for detecting overlapping communities [37, 38, 39, 40]. CONGA [37] detects overlapping communities by removing the edge or splitting the vertex with the highest betweenness centrality value. CONGO [38] is a fast version of CONGA [37] by calculating local betweenness centrality instead of global one. The algorithm proposed in [39] is based on a label propagation technique, and the algorithm proposed in [40] is designed for detecting fuzzy overlapping communities. Moreover, due to the fact that overlapping is a nature property of graph data, more papers are published on this topic. Coscia et al. [21] proposes an algorithm by extracting redundant subgraphs by defining graph operations. Label propagation algorithm is performed on these subgraphs to obtain overlapping communities. Yang and Leskovec [121] propose a scalable algorithm to detect overlapping communities of networks with millions of nodes and edges. In Chapter 6, we propose a attributed graph clustering algorithm which is able to achieve overlapping clusters detection.

Bipartite Graph Clustering

Graph becomes more complex with the development of information acquisition. Although bipartite graph can be transferred into simple graph, there will be information loss during the process. Moreover, we prefer acquiring the relationship between multiple type of vertices as well. In real word, many data can be modeled as bipartite graphs in order to pursuit the better description of the relationship between two type of objects, for example customs and items data, gene and protein data. Therefore, bipartite graph clustering is defined in Definition 2.4, in which clusters only exist on the same type of vertices.

Definition 2.4 (Bipartite Graph Clustering) *Given a bipartite graph $G = (V_1, V_2, E)$ with V_1 and V_2 representing two type of the vertics and E standing for edges between two types of vertics, bipartite graph clustering divides V_1 into K components $\{C_{11}, C_{12}, \dots, C_{1K}\}$ and V_2 into L components $\{C_{21}, C_{22}, \dots, C_{2L}\}$, which fulfills the conditions that $C_{11} \cup C_{12} \cup \dots \cup C_{1K} = V_1$ and $\forall i \neq j, C_{1i} \cap C_{1j} = \emptyset, i = \{1, 2, \dots, K\}, j = \{1, 2, \dots, K\}$, $C_{21} \cup C_{22} \cup \dots \cup C_{2L} = V_2$ and $\forall i \neq j, C_{2i} \cap C_{2j} = \emptyset, i = \{1, 2, \dots, L\}, j = \{1, 2, \dots, L\}$.*

When facing such data matrix, traditional algorithms will suffer from the high dimensional problem. Bipartite graph clustering also named as bi-clustering or co-clustering which simultaneously clusters rows and columns of a data matrix is able to avoid the curse of dimensionality. Meanwhile, it is able to analysis the relationship between rows and columns as well. In recent years, co-clustering or bi-clustering has received lots of attention and many papers have been published. First of all, Dhillon et al. have proposed many state-of-the-art co-clustering algorithms and has applied these algorithm to many real life fields [23, 25, 20, 5]. Specifically, they proposed a spectral algorithm for bipartite graph partition in [23]. Information-theoretic Co-clustering [25] adopts mutual information to measure information loss. The best clustering result owns the smallest information loss. Bregman divergences based co-clustering [5] aims to achieve clustering of rows and columns by searching for a good matrix approximation. Bregman divergences is adopted to measure the approximation error. Information-theoretic Co-clustering and Bregman divergences based co-clustering both need to predetermine the number of cluster for each type of nodes. Besides both of them are not confined to deal with binary matrix. Long et al. [71, 70] formulate co-clustering as an optimization problem of matrix decomposition, but they still need to set the number of clusters of rows and columns. In case of real world data, Cho et al. [20] apply co-cluster to gene express data and Dhillon [23] use it to effectively process documents and words data. Moreover, Cross-association [17] is a parameter-free co-clustering method based on Minimum Description Length principle. Information-theoretic co-clustering [25] and Cross-association [17] are used as the comparison methods of our proposed algorithm SCMiner [32], and details can be seen in Chapter 4. In addition, there are still many other co-clustering algorithms, for example [96, 36].

Complex Graph Clustering

As social network topic is becoming a hot topic, the construction of graphs is not limited to single type of vertices and edges. In such complex graph, clusters are defined as group of vertices or edges sharing the same properties. For example, a simple publication data contains three types of vertices (author, paper, venue) and two types of links (writing, publishing). Then the clusters

of such data can be constructed by authors, papers and venues which belong to the same research field.

In recent years, many algorithms are designed to solve the complex graph clustering problem. Sun and Han define a Heterogeneous Information Network (HIN) [106] which abstracts the real social network data containing nodes and links with different types. Then they propose algorithms named Rankclu [107] and Netclus [108] which adopt the idea of integrating clustering and ranking to detect clusters from Heterogeneous Information Network. Additionally, Sun et al. [105] proposes a model-based method for clustering Heterogeneous Information Networks containing incomplete attributes. Moreover, another type of complex network named as multiple graph that contains the same set of vertices and heterogeneous edges relations also frequently appears in literatures. The key procedure for clustering such type of graphs is integrating heterogeneous edge relations to detect clusters of vertices. For instance, Tang et al. [109] achieve clusters by fusing multiple link information based on matrix factorization. For some more complex social media data, Lin et al. [68] propose a novel hypergraph representation model named as metagraph which can represent multi-relational links, multiple type of vertices and time-varying relations.

2.1.4 Graph Structure Mining

Another important graph mining task relevant to the dissertation is graph structure mining. Different links lead to various graph topological structures. Thus structure mining is a meaningful task in acquiring knowledge from graph data. In the following, we analyze the global graph structure and local graph substructures respectively.

Topological Structure Mining

In network analysis, certain network data has a specific topology. For example, social network has the feature that most of the nodes are connected by a small number of links. And this feature is named as small world phenomenon. Wang and Chen [116] point out several types of typical complex network models and their properties. Based on the basic graph properties mentioned

above, network can be categorized into different models. To be specific, ER random graph [28] is the graph that each pair of vertices has equal probability p to be connected. Watts and Strogatz [117] introduce the small-world network which has small average shortest path length and high clustering coefficient. Due to the high clustering coefficient, cliques and near-cliques frequently appear in small-world networks. Many real-world networks show small-world phenomena, such as friendship networks, road maps, food chains, electric power grids, metabolite processing networks and networks of brain neurons. Scale-free network [4] has a power-law degree distribution. Nodes with large degree named as hubs are normally be seen in such type of network. Many real-world networks are scale-free network. For example, collaboration networks, world wide web and Protein-protein interaction networks.

Substructure Mining

Substructure is the subset of a graph. Substructure mining aims to discover graph structures that occur frequently or to detect the graph structures that are interesting. The Apriori-based algorithm AGM [52] efficiently mines the frequent substructures based on association rule. It generates candidate substructures by level-wise search. FSG [64] is another Apriori-based algorithm that uses sparse graph representation to store candidates in order to speed up computation. Yan and Han [120] propose a pattern growth approach gSpan which adopts depth-first search to discover frequent subgraph without considering candidates generation and false positive pruning. SUBDUE [50] is an graph-based knowledge discovery system based on Minimum Description Length principle. It discovers frequent substructure by compressing the patterns and selects the one with minimum bits. The above example algorithms all consider frequently appeared patterns in the graph as interesting. Moreover, there are some other algorithms proposed for detecting interesting structures. For instance, Oddball [2] is a technique to identify interesting near-cliques and star-like structures by extracting features to represent the nodes of a graph in a low-dimensional vector space. RolX [48] is an unsupervised learning algorithm for automatically discovering structural roles of nodes by adopting feature extraction, feature grouping and model selection.

2.1.5 Link Prediction

Link prediction is a significant edge related task in graph mining. Link prediction is generally classified into two types: one is predicting missing links in a static network, the other is predicting the links which will appear in the future. Link prediction techniques are widely applied in real network. For example, social network websites like Facebook often recommend friends or interesting topics to the users. In some biological networks like protein-protein interaction networks, we are able to predict which pairs of proteins are most likely to have close relation thus saving experimental cost.

In the view of the existing link prediction algorithms, these algorithms are divided into unsupervised algorithms [66] and supervised algorithms [45, 57, 62, 67].

For the unsupervised algorithms, Liben-Nowell and Kleinberg [66] summarize some basic link prediction scores. To be specific, we can give each pair of disconnected nodes a score, and then sort these pairs of nodes in descending order based on the scores. The pair of nodes taking the head place has the highest probability becoming an edge in the future. Following we will introduce some classical measures.

Common Neighbors. Common neighbors [75], as shown in Eq. (2.11), is an intuitive method which calculates the number of overlapping neighbors that node x and node y share, where $\Gamma(x)$ denotes the set of neighbors of node x .

$$Score_{CN} = |\Gamma(x) \cap \Gamma(y)|. \quad (2.11)$$

Jaccard's Coefficient. Jaccard's Coefficient [93] is shown in Eq. (2.12), which is based on common neighbors but has a value between 0 and 1.

$$Score_{Jaccard} = \frac{|\Gamma(x) \cap \Gamma(y)|}{|\Gamma(x) \cup \Gamma(y)|}. \quad (2.12)$$

Adamic/Adar. Adamic/Adar [1] measures the similarity of two node. Eq. (2.13) considers common neighbors as features and assigns more weight to neighbors that are not shared with

many others.

$$Score_{AA} = \sum_{z \in \Gamma(x) \cap \Gamma(y)} \frac{1}{\log|\Gamma(z)|}. \quad (2.13)$$

Preferential Attachment. Preferential attachment [77] is introduced in network generation of power law distributions. As a link prediction score as shown in Eq. (2.14), it holds the meaning that nodes are preferring to connect with the node with the high degree.

$$Score_{PA} = |\Gamma(x)| \cdot |\Gamma(y)|. \quad (2.14)$$

Katz. Katz [58] is a shortest path based score measure, which is shown in Eq. (2.15), where l is the length of the path, $paths_{x,y}^{<l>}$ is the path between node x and node y with the length equals to l , β is the factor which is adopted to penalize the contribution of longer paths. It can be also adopted in weighted graph by considering the number of times of the path $paths(x, y)$.

$$Score_{Katz} = \sum_{l=1}^{\infty} \beta^l \cdot |paths_{x,y}^{<l>}|. \quad (2.15)$$

There are also many other scores, like PageRank [13], SimRank [53], which are also effective in predicting the links.

Supervised link prediction approaches are treated as a binary classification problem. Various node pair features like topological features or attributes similarities can be extracted. Various supervised learning algorithms like decision tree, K-nearest neighbors or support vector machines (SVM) can be used on the features to distinguish which node pairs will be connected in the future. Treating link prediction as the classification suffers the class skewness problem due to the very large size of possible links.

Besides, in recent years, there are many other algorithms proposed to solve the link prediction problem. For example, Wang et al. [115] use a MRF based local probabilistic model and Kashima and Abe [57] use a parameterized probabilistic model.

2.1.6 Graph Compression

Graph data is normally large and massive, which leads graph mining facing new challenges. Intuitively, graph compression is an effective method for solving the graph storage problem. Boldi and Vigna [10] propose a web graph compression scheme which is able to effectively store graph. Motivated by [10], Chierichetti et al. [19] proposes a compression strategy for social network. Moreover, graph compression is also an effective method to further understand the graph structure. Graph summary aims to group the similar nodes or edges in order to simplify the graph. For example, Navlakha et al. [81] propose graph summarization algorithms by greedy merging the original graph to a summary graph and a correction matrix. Tian et al. [110] propose a graph summarization algorithm by grouping nodes based on their attributes and relationships which are selected by the user. Additionally, graph compression is a technique adopted to obtain other knowledge. For example, SUBDUE [50] discovers frequent substructure by compressing the patterns and selects the one with minimum bits. Cross-association [17] is a compression based parameter-free co-clustering method that detects clusters of rows and columns alternately. Mueller et al. [80] propose an clustering algorithm based on weighted graph compression. SLASHBURN [54] uses the power-law characteristic for compression and exploits the hubs and the neighbors of hubs.

2.2 Basic Knowledge of Information Theory

2.2.1 Information Theoretic Concepts

First of all, let us start with the introduction of information theory. Briefly speaking, information theory is a theory considering “information” as research object. What is information? As the core concept, information is broadly expressed as the content and the form of the message, such as the emails we have received, the sounds we have heard and the videos we have watched. But in a narrow sense, information refers to statistic information which concerns more about the expression form than the content of the information. For example, information is expressed as signal and is transferred from sender to receiver. Information theory is formed based on the narrow con-

cept by Claude E. Shannon. His famous paper “A Mathematical Theory of Communication” [97] published in 1948 is considered as a cornerstone of information theory. At the beginning, information theory is founded on communication system, then it is extended to mathematic, computer science and electronic engineering. It emphasizes on researching communication system, data transmission, data compression by probability theory and statistics.

Shannon points out that information is uncertain and he defines entropy, joint entropy, conditional entropy and mutual information as important measures of information.

Entropy. First of all, we introduce entropy [51] which is adopted to quantify the uncertainty of the information, which is defined by Eq. (2.16). In this equation, X represents a random variable which contains n components $\{x_1, x_2, \dots, x_n\}$ with the probabilities denoted as $\{p(x_1), p(x_2), \dots, p(x_n)\}$. In the thesis, we select the base of the logarithm to 2, which shows that the information is represented by bits. In other words, the description of information adopts a string of '0's and '1's. High entropy indicates that the random variable is unpredictable because of the balanced distribution of each component and the random variable dominated by some certain components provides low entropy, having a high level of predictability.

$$H(X) = - \sum_{i=1}^n p(x_i) \cdot \log_2 p(x_i). \quad (2.16)$$

Joint Entropy. Joint entropy and conditional entropy is generated from basic entropy. Joint entropy measures uncertainty of multiple variables. Eq. (2.17) shows the joint entropy of two random variables X and Y , $\{x_1, x_2, \dots, x_n\}$ and $\{y_1, y_2, \dots, y_m\}$ are their components respectively. Due to joint entropy $H(X, Y)$ measures uncertainty of X and Y , $p(x_i, y_j)$ is the probability of component x_i and y_j occurring together.

$$H(X, Y) = - \sum_{i=1}^n \sum_{j=1}^m p(x_i, y_j) \cdot \log_2 p(x_i, y_j). \quad (2.17)$$

Conditional Entropy. The conditional entropy of X and Y is calculated in Eq. (2.18) and Eq. (2.19). $H(Y | X)$ is the entropy of Y under the condition of X , which measures uncertainty of the random variable Y based on the condition that the random variable X is known. Likewise,

$H(X | Y)$ is the conditional entropy of X . And Eq. (2.20) shows the relationship of joint entropy and conditional entropy. When the entropy of random variable X or Y is known, we only need to calculate the conditional entropy of the other unknown random variable Y or X and the joint entropy of the X and Y can be obtained.

$$H(Y | X) = - \sum_{i=1}^n \sum_{j=1}^m p(x_i, y_j) \cdot \log_2 \frac{p(x_i)}{p(x_i, y_j)}. \quad (2.18)$$

$$H(X | Y) = - \sum_{i=1}^n \sum_{j=1}^m p(x_i, y_j) \cdot \log_2 \frac{p(y_j)}{p(x_i, y_j)}. \quad (2.19)$$

$$\begin{aligned} H(X, Y) &= H(X | Y) + H(Y) \\ &= H(Y | X) + H(X). \end{aligned} \quad (2.20)$$

Mutual Information. Mutual Information measures how much information the two random variables X and Y share, which is shown in Eq. (2.21). And mutual information can be also calculated from joint entropy and conditional entropy, which is shown in Eq. (2.22).

$$I(X, Y) = \sum_{i=1}^n \sum_{j=1}^m p(x_i, y_j) \cdot \log_2 \frac{p(x_i, y_j)}{p(x_i)p(y_j)}. \quad (2.21)$$

$$\begin{aligned} I(X, Y) &= H(X) - H(X | Y) \\ &= H(Y) - H(Y | X) \\ &= H(X) + H(Y) - H(X, Y) \\ &= H(X, Y) - H(X | Y) - H(Y | X). \end{aligned} \quad (2.22)$$

2.2.2 Minimum Description Length Principle

Data compression is the technique that uses less bits to transmit or store original data by specific encoding mechanism. Data compression is based on the fact that data is biased. For example, a data string “1000101011” is constructed by ten binary figures which is generated randomly. There is no regularity can be found, thus it is hard to compress such unbiased data. The other

data string ‘1010101010’ is also a ten binary figure string which contains five times repetition of ‘10’. In such cases, we can compress the data by the regular pattern of five times ‘10’. These are two simple extreme examples. Normally, data can be compressed by its regularities.

Coding methods are the key technique for data compression. Coding essentially maps original data to finite binary string. For example, Huffman Coding is a lossless entropy encoding method. It assigns symbols with variable-length code and the length of the code is determined by the probabilities of the symbols. The symbol with high probability will be assigned with short code and vice versa. That is to say, data compression by Huffman Coding saves bits by assigning frequent symbols with shorter code. For example, there is a data string ‘aababca’, ‘a’ is the most frequently appeared symbol and is assigned with the shortest code ‘0’; the symbol ‘b’ appears two times and is represented with longer code ‘01’; the symbol ‘c’ only appears one time and is assigned with the longest code ‘110’. The length of the code is decided by the frequency of the symbol.

Data compression is categorized into lossless compression and lossy compression. As its name implies, lossless compression is the technique which can reconstruct the original data exactly from the compressed data, while lossy algorithms can only reconstruct an approximation of the original data from the compressed data. In the thesis, we adopt Minimum Description Length [92] (MDL) principle, a lossless compression method, as the key technique.

MDL is first proposed by Jorma Rissanen in 1978 [90], which follows the assumption that the less coding length we adopt to describe the data, the more knowledge we can gain from it. Formally, the quality of a model can be identified from Eq. (2.23), where $L(M)$ denotes the coding length for describing model M and its parameters, while $L(D | M)$ represents the cost of coding data D under model M . MDL balances the model and the data and aims to find the best result.

$$L(M, D) = L(D | M) + L(M) \quad (2.23)$$

MDL only considers the length of the codes and do not care about the content of the codes. The key task of adopting MDL is how to design the coding scheme for the data. Normally, we estimate the probability distribution $P(x_i)$ of the random variable $X = \{x_1, x_2, \dots, x_n\}$, then the

coding length of X can be constructed as Eq. (2.24), where $L(X)$ is the coding length of the random variable.

$$L(X) = - \sum_{i=1}^n \log_2 P(x_i). \quad (2.24)$$

MDL is a method for model selection, prediction, parameter estimation, etc. For example, clustering can be regarded as a model selection problem considering clustering assignments as the model. According to finding trade-off between model and the data, the optimal clustering results can be obtained. In this thesis, we proposed several algorithms based on MDL principle to deal with graph mining problem. In these algorithms, we are able to detect clusters of the graph, find unique structure or predict the links. With the aid of the MDL principle, we do not need to set any parameters, e.g. the number of the clusters. We can also avoid the over-fitting problem with too complex model.

2.3 Evaluations in Graph Mining

Many algorithms have been proposed to deal with various tasks of graph mining. In this section, we will introduce the methods for evaluating the algorithms that we used in the thesis for graph clustering and link prediction.

Purity. Graph clustering assigns each vertex with only one or multiple labels. Comparing with the given standard labels, we are able to adopt the metrics which are described below to evaluate the clustering results. Set $C = \{C_1, C_2, \dots, C_K\}$ is the clusters which are obtained by the algorithms and $L = \{L_1, L_2, \dots, L_M\}$ is the true labels of the data set. Then purity [126] of each cluster can be calculated by Eq. (2.25), which measures the percentage of dominating elements. The larger the purity is, the better the quality of the cluster is.

$$Purity(C_i) = \frac{1}{|C_i|} \max_j |C_i \cap L_j|. \quad (2.25)$$

NMI, AMI and AVI. Normalized mutual information (NMI) [103], adjusted mutual informa-

tion (AMI) [114] and Adjusted Variation of Information AVI [114] are information theoretic based measure. Normalized mutual information (NMI) can be calculate from Eq. (2.26), where $I(X, Y)$ is mutual information which is shown in Eq. (2.21), $H(X)$ and $H(Y)$ are entropy of random variables X and Y which can be calculated from Eq. (2.16). To be specific, suppose that acquired cluster label $C = \{C_1, C_2, \dots, C_K\}$ and true label $L = \{L_1, L_2, \dots, L_M\}$ are two random variables. Thus NMI can be calculated from Eq. (2.27), where N is the number of all the objects, N_{C_i} is the number of objects in cluster C_i , N_{L_j} is the number of objects in the true label class L_j and N_{C_i, L_j} is the number of objects in both cluster C_i and true label class L_j .

$$NMI(X, Y) = \frac{I(X, Y)}{\sqrt{H(X)H(Y)}}. \quad (2.26)$$

$$\phi^{NMI}(C, L) = \frac{\sum_{i=1}^K \sum_{j=1}^M N_{C_i, L_j} \log\left(\frac{N \cdot N_{C_i, L_j}}{N_{C_i} N_{L_j}}\right)}{\sqrt{\left(\sum_{i=1}^K N_{C_i} \log \frac{N_{C_i}}{N}\right) \left(\sum_{j=1}^M N_{L_j} \log \frac{N_{L_j}}{N}\right)}}. \quad (2.27)$$

Similarly, adjusted mutual information (AMI) and adjusted variation of information (AVI) are shown in Eq. (2.28) and Eq. (2.29) respectively, where $E(I(X, Y))$ is the expected value of mutual information $I(X, Y)$ between all possible pairs of clusters.

$$AMI(C, L) = \frac{I(C, L) - E\{I(C, L)\}}{\sqrt{H(C)H(L) - E\{I(C, L)\}}}. \quad (2.28)$$

$$AVI(C, L) = \frac{2I(C, L) - 2E\{I(C, L)\}}{H(C) + H(L) - 2E\{I(C, L)\}}. \quad (2.29)$$

All the NMI, AMI and AVI value are bounded in $[0, 1]$, the higher the value is, the better the clustering results are.

Precision, Recall and F-Measure. Precision [89] measures the accuracy of the detected clusters and recall [89] measures whether all clusters are detected. Precision and recall can be calculated from Eq. (2.30) and Eq. (2.31) separately. In the equations, TP is True Positive, FP is False Positive and FN is False Negative. Comparing with clusters C and true labels L , True Positive is the number of pairs of objects which appear in the same cluster in both clusters C and true labels

L , False Positive is the number of pairs of objects which appear in the same cluster in clusters C but in different class of true labels L , False Negative is the number of pairs of objects which appear in the different cluster in both clusters C and true labels L .

$$Precision = \frac{TP}{TP + FP}. \quad (2.30)$$

$$Recall = \frac{TP}{TP + FN}. \quad (2.31)$$

$$F - Measure = (1 + \beta^2) \frac{Precision \cdot Recall}{\beta^2 \cdot Precision + Recall}. \quad (2.32)$$

F-Measure [89] is shown in Eq. (2.32) which combines precision and recall. And β is a non-negative value weighting the precision and recall, which usually equals 0.5, 1 and 2.

Precision for Link Prediction. In the thesis, precision [72] is adopt to evaluate the result of link prediction methods. A certain percentage of edges are selected as predicting edge PE and are deleted from the graph. Algorithms are implemented on the new formed data and predicted edges RE are acquired. Comparing with predicting edge PE and predicted edges RE , the precision of link prediction can be calculated from Eq. (2.33).

$$Precision_{LP} = \frac{|PE \cap RE|}{|PE|}. \quad (2.33)$$

Chapter 3

Automatically Spotting Information-rich Nodes in Graphs

In this thesis, we start with mining knowledge from the simple graph. We focus on a novel graph mining task for detecting interesting nodes from a simple graph. What kind of nodes are interesting in a graph? When we talk about nodes detection, the first impression is that the problem seems similar with the outlier detection in vector data. However, following the definition of outlier in vector data, we will find isolated nodes which are not interesting in the graph. On the other hand, interesting nodes can be judged by the background knowledge. For example, given the job information, we can easily distinguish leader nodes and staff nodes. However, the background knowledge is not always available. Normally, the graph only provides structural information. Therefore, it is necessary to propose a method which is able to detect interesting nodes based on the structural information. In this chapter, we propose an algorithm named as Info-spot which detects interesting nodes from the perspective of link patterns. The algorithm is proposed based on the information theoretic knowledge, especially the Minimum Description Length (MDL) principle. Guided by MDL, nodes exhibiting similar link patterns are greedily merged, then interesting nodes naturally emerge. Meanwhile, Info-spot is a fully automatic algorithm which overcomes the parameters problem that most existing nodes detection approaches suffer. Finally, the experiments demonstrate the benefits of our approach.

The remainder of this chapter is organized as follows: In Section 3.1, it starts with an introduction. Section 3.2 defines the information content of a node. Section 3.3 introduces the similarity measure. Section 3.4 combines similarity measure with embedding. Section 3.5 presents the algorithm Info-spot in detail. Section 3.6 describes the experiments. Section 3.7 discusses related work and Section 3.8 gives the conclusion of this chapter.

Parts of the material presented in this chapter have been published in [47], where Jing Feng was responsible for the development of the main concept, implemented the main algorithms and wrote parts of the paper; Xiao He helped with the development of the main concept, performed parts of experiments and wrote part of the paper; Claudia Plant supervised the project and revised the whole paper.

“Xiao He, Jing Feng, Claudia Plant. Automatically Spotting Information-Rich Nodes in Graphs. The IEEE ICDM 2011 Workshop on Data Mining in Networks DaMNet: 941-948.”

3.1 Introduction

3.1.1 Motivation

In many applications including, e.g., social media, network traffic and protein interaction data, the detection of outstanding and interesting data objects is the most important data mining goal. Especially for vector data, the research area of outlier detection has therefore attracted much attention and is very mature with multiple research papers, surveys and books, e.g., [59, 12, 9], to mention a few. Although large graphs are prevalent in nature, consider, e.g., social and publication networks, road networks and biological pathways, only disproportional few approaches, such as [2, 27, 84] focus on the detection of outstanding nodes in graphs. The traditional outlier notion for vector data cannot be straightforwardly extended to graph data but needs to be completely redefined. For vector data, three basic approaches to outlier detection have been proposed: parametric statistical methods, distance-based and density-based data mining techniques.

All these approaches have in common that they search for data points which are far away from the rest of the data set.

Translating this outlier notion directly to graph mining would mean to search for isolated nodes. At first glance, this perhaps seems to be easy. However, most real-world graphs are very sparse and characterized by a small-world organization. Although most nodes are connected to very few other nodes, almost every pair of nodes is connected by a short path. It is therefore difficult to distinguish between outstanding and ordinary nodes based on isolatedness. Moreover, consider the extreme case of a completely isolated node forming a singleton connected component. Such a node would be rated as very outstanding but is not interesting at all for knowledge discovery. The reason for this is evident from Hawkins' fundamental definition [46]: An outstanding observation can only be interpreted in the context of the other observations in the data set. Since a singleton node is not connected to the rest of the graph, it does not represent any interesting information for interpretation. Therefore, in contrast to vector data, isolatedness is not a helpful property to detect outstanding nodes in a graph. Outlier detection on graphs requires completely novel outlier notions. An interesting recent approach, Oddball [2] proposes techniques for feature extraction to represent the nodes of a graph in a low-dimensional vector space. Interesting near-cliques and star-like structures can be identified with standard outlier detection techniques for vector data. Note that in Oddball outlying nodes are not isolated nodes but rather central nodes in extraordinary subgraphs.

3.1.2 Contributions

In this chapter, we focus on automatically detecting interesting nodes in large graphs. To quantify the interestingness of a node, we consider the information content of its link pattern from the perspective of data compression. To illustrate the basic idea of information theoretic for the detection of interesting node, let us take data transmission for example. We want to transfer the information of a graph-structured data set via a communication channel from a sender to a receiver. Normally, we need to transfer the full adjacency matrix, requiring communication costs proportional to its entropy. However, most natural graphs can be much more effectively

compressed since they contain regularities: many objects approximately follow some typical link patterns. Only some very interesting nodes exhibit a special and unique link patterns which cannot be effectively compressed with any of the ordinary link patterns.

Figure 3.1 illustrates an intuitive example: To transfer a graph data G , a naive way would be to directly compress the adjacency matrix of G (the top path in Figure (3.1)), the coding costs are 20806 bits in total. However, this graph exhibits strong regularities which could be used to improve compression. In particular, there are two groups of vertices of size 100 (dark blue nodes) and 49 (pink nodes) having exactly same link patterns. We thus group them together and only transfer the adjacency matrix of a four-node graph G_R and spend some additional bits to store the group information (the bottom path in Figure (3.1)). This simple coding scheme only requires about 168 bits in total, where the drastically reduced adjacency matrix requires about 14 bits and the cluster ID required around 154 bits. Finally, the light blue vertex with ID 3 and red one with ID 4 can be easily labeled as outstanding compared with the other nodes, since they have different link patterns.

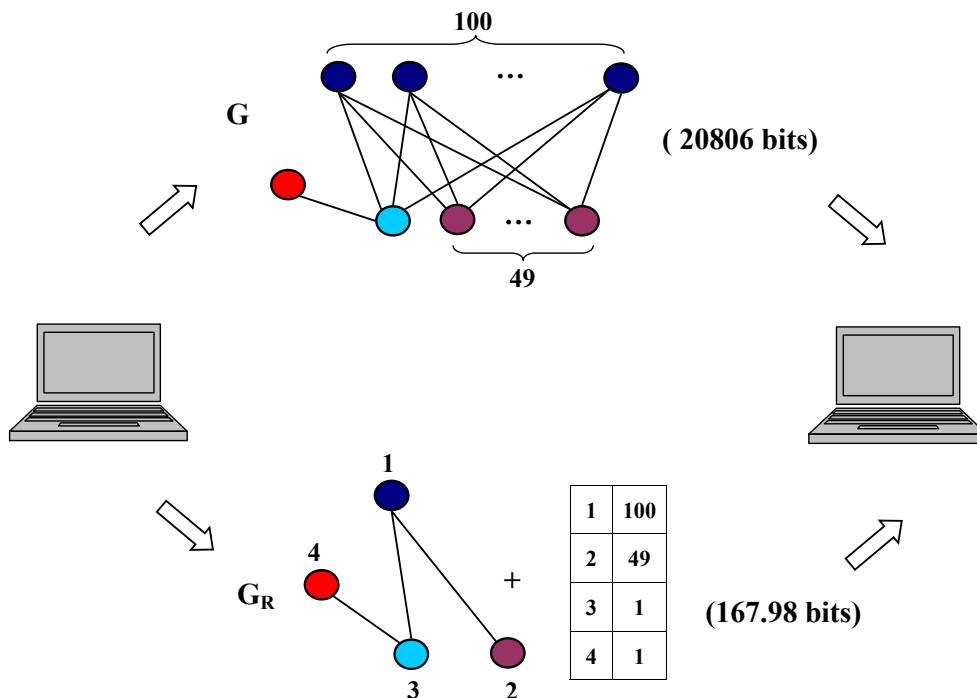


Figure 3.1: An Example of Using Regularities to Compress Graph and Detect Outstanding Nodes

The major contributions of our approach can be summarized as follows:

- **Information content as a novel outlier notion for graphs.** The information content of the link pattern is an intuitive and flexible way of quantifying the interestingness of a node. This novel outlier notion allows detecting a wide range exceptional link patterns, including e.g., centers of star-like structures and hubs.
- **An intuitive similarity measure for comparing nodes.** We formalize this idea to an intuitive similarity measure quantifying the amount of different information in bits between the link patterns of nodes.
- **Algorithm Info-spot for fully automatic spotting interesting nodes.** Based on the Minimum Description Length (MDL) principle [41], our novel algorithm Info-spot automatically flags the most interesting nodes of a graph without requiring any input parameters.

3.2 The Information Content of A Node

3.2.1 Naive Coding Scheme

We consider an unweighted and undirected graph G with vertices $V = \{A, B, C, \dots\}$ denoted by capital letters and edges $E = \{AB, AC, \dots\}$ labeled with the incident nodes, $|E|$ denoted the number of edges and $|V|$ denoted the number of nodes.

The most direct and simple way to design a coding scheme to compress a graph G is to encode its adjacency matrix $\mathcal{A} \in |V| * |V|$, with $\mathcal{A}_{i,j} = 1$ if $ij \in E$. The coding cost of the adjacency matrix of the graph G is lower bounded by its entropy which is provided by:

$$CC(G) = |V|^2 \cdot H(G) \quad (3.1)$$

where $H(G) = -(p_1(G) \cdot \log_2 p_1(G) + p_0(G) \cdot \log_2 p_0(G))$, $p_1(G)$ and $p_0(G)$ are the probabilities of 1 and 0 in the adjacency matrix of graph G . Although the naive coding scheme is very simple, its effectiveness is far from the optimal compression rate. This is because of the underlying

assumption that each cell of the adjacency matrix is independent from each other cell, which is the case in a randomly generated graph. Most real-world graphs however, are not random but have been generated by some distinct statistical processes which cause dependencies among the link patterns of the vertices.

3.2.2 Information Content of A Node

In adjacency matrix, the link pattern of a vertex can be written in a binary vector, where ‘1’ indicates its connection and vice versa. This well captures the relationship between the vertex and all the other vertices, which means that it contains all the information of a node in a graph. Therefore, we define the information content of the link pattern to quantify the interestingness of a node.

Definition 3.1 (Information Content of a Node) *The link pattern of a node v is a vector $L_v = \{l(1), l(2), \dots, l(i), \dots, l(|V|)\} \in |V| \times 1$. $l(i) = 1$ if node v has edge with node i . Otherwise $l(i) = 0$. We define the information content of a node as the amount of bits coding its link pattern:*

$$I(v) = I(L_v) = \begin{cases} |V| \cdot H(L_v) & \text{if } 0 < p_1 < 1 \\ 1 + \log_2 |V| & \text{else} \end{cases} \quad (3.2)$$

where $H(L_v) = -(p_1(L_v) \cdot \log_2 p_1(L_v) + p_0(L_v) \cdot \log_2 p_0(L_v))$, where $p_1(L_v)$ and $p_0(L_v)$ are the probabilities of ‘1’ and ‘0’ in the link pattern of node v . When $p_1(L_v) = 0$ or $p_0(L_v) = 0$, which means that the vertex is isolated or it connects with all the other vertices, the entropy of the vertex’s link pattern is 0. However we still need some bits to transfer the link pattern. As Eq. (3.2) showed, 1 bit is used to code whether the vertex is isolated or connected to all the other vertices, and $\log_2 |V|$ is the bits to code the length of pattern.

3.3 Similarity Measure

In this section, we introduce an information-based similarity measure between the link patterns of two vertices and exploit this similarity for effective graph compression. When encoding the

adjacency matrix of a graph, we can also consider calculating the information content of each vertex separately and sum up over all vertices. With this vertex-centered view we can not only quantify how many bits we need to encode that particular vertex and its link pattern, i.e. its incident edges. In particular, we can also compare two vertices A and B by quantifying the amount of non-redundant information between their link patterns. If A and B have very similar link patterns, which means that they are connected to similar vertices, instead of encoding the link patterns of A and B separately, we can improve the graph compression if we only encode A with its information content and additionally specify a small amount of relational information to represent B .

To calculate the relational information of two nodes, we first define a relation pattern of them, then with the information content of the relation pattern we get the similarity.

Definition 3.2 (Relation Pattern Between Two Nodes) *The link pattern of node A and node B are L_A and L_B , the relation pattern between A and B is R_{AB} , with*

$$R_{AB}(i) = \begin{cases} 1 & \text{if } L_A(i) = L_B(i) \\ 0 & \text{if } L_A(i) \neq L_B(i) \end{cases} \quad (3.3)$$

Figure 3.2 is an example of the relation pattern of two nodes A and B .

L_A	1	0	1	1	0
L_B	1	1	0	1	0
R_{AB}	1	0	0	1	1

Figure 3.2: Relation Pattern of Two Nodes

The relation pattern of two nodes can be quantified in amount of bits according to the information content of a pattern defined in Eq. (3.2), similarly we define the bits of coding the relation pattern of two nodes to be their similarity with

$$I(R_{AB}) = \begin{cases} |V| \cdot H(R_{AB}) & \text{if } 0 < p_1 < 1 \\ 1 + \log_2 |V| & \text{else} \end{cases} \quad (3.4)$$

where $H(R_{AB})$ is the entropy of the relation pattern. From this definition we can get an intuitive similarity measure. The smaller the information content of relation pattern of two nodes is, the more similar they are.

As mentioned before, we can improve graph compression using proposed relation pattern and similarity. Take Figure 3.2 as an example, if we encode the link pattern of nodes A and B separately to represent them, the coding cost is $I(L_A) + I(L_B) \simeq 9.7$ bits, but if we only use the link pattern of node A and relation pattern of them for coding, the cost is $I(L_A) + I(R_{AB}) \simeq 8.5$ bits, which is smaller. Such strategy can help us to find the optimal coding of a graph, and further get the interesting nodes.

3.4 Embedding

In this section, we combine the proposed similarity measure with metric embedding to verify its effectiveness. The proposed matrix similarity can be regarded as a high dimensional data ($|V| \times |V|$) with $|V|$ observation and $|V|$ dimension. Therefore, we can directly use any mature outlier detection method for vector data by mapping the similarity matrix to a lower dimensionality. Multidimensional Scaling (MDS) [11] is an efficient and effective algorithm to find a low-dimensional embedding of a similarity matrix. However, it is hard to decide how many dimensions should we choose to map the similarity matrix onto. Closely related to MDL, the Bayesian Information Criterion (BIC) [65] is a method to decide a suitable dimensionality for embedding. BIC is defined as follows:

$$BIC = \sum_{i < j} (d_{ij} - \hat{d}_{ij}) + m \cdot n \cdot \log_2 \frac{n(n-1)}{2}, \quad (3.5)$$

where d_{ij} is the distance of any two entities in similarity matrix, \hat{d}_{ij} is distance of any two entities in embedding; n is the dimensionality of the similarity matrix, in our case, it corresponds to the number of vertices; m is the dimensionality of the embedding. We compute the BIC for each value of m , which ranging from $1, 2, \dots, n$ and choose the minimum as suitable dimensionality for embedding. The experiment results show the effectiveness of proposed similarity, which are

depicted in Figure 3.6 in Section 3.6.

3.5 Algorithm Info-spot

In this section we first introduce a MDL-based coding scheme, then based on this coding scheme we present Info-spot, a greedy algorithm which iteratively merges the two most similar vertices based on the proposed similarity measurement until all vertices are merged together, and finally outputs the graph representation with the minimum MDL. All nodes remaining isolated can be easily flagged as interesting.

3.5.1 MDL-based Coding Scheme

The Minimum Description Length (MDL) principle has recently attracted much attention for model selection for many data mining tasks including classification [113] and clustering [8, 7]. It is based on the following insight: Any regularity in the data can be used to compress it and the more we are able to compress the data, the more we have learned about it [41]. Therefore, we can get the best graph representation when we get the MDL representation of the graph.

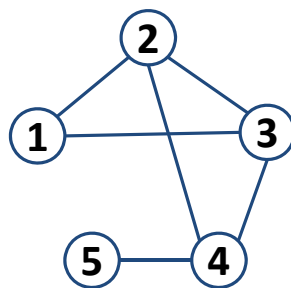


Figure 3.3: An Example of Graph

As mentioned in previous section, we can represent a graph by compressing its adjacency matrix, but if there exists one pair of vertices having similar link patterns, we can merge them together and using the relation pattern of them and the adjacency matrix of remaining vertices to represent the graph instead. The following equation defines the coding cost of a graph using

such representation method:

$$CC(G) = CC(G_R) + I(R) + 2 \cdot \log_2 |V| \quad (3.6)$$

where G_R is the graph with remaining vertices, $CC(G_R)$ is the naive coding cost of G_R , $I(R)$ is the coding cost of relation pattern between two merged vertices, $2 \cdot \log_2 |V|$ is the bits to code the ID of merged nodes.

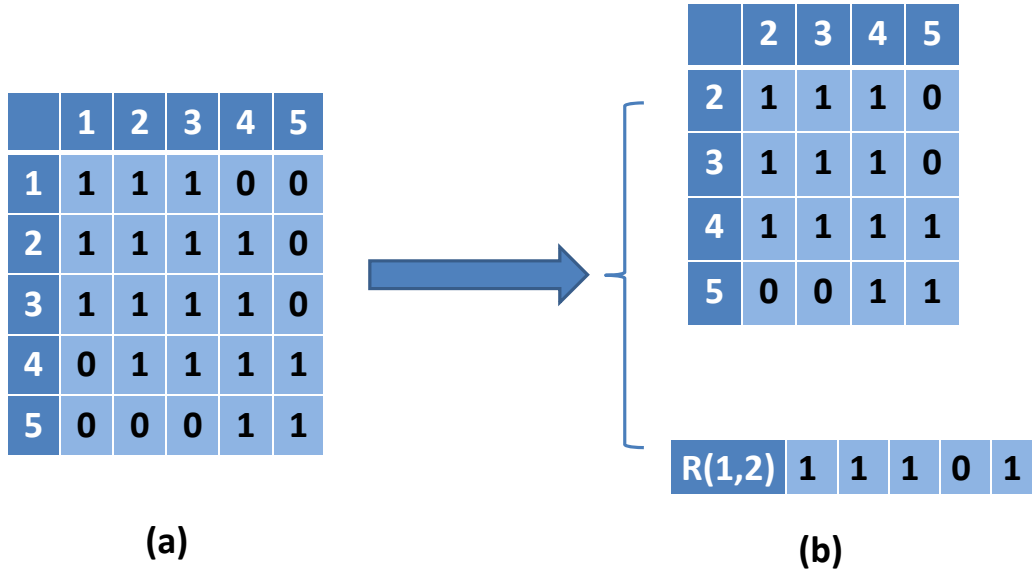


Figure 3.4: MDL Coding Example

Figure 3.3 displays a simple graph which we use in the following. Figure 3.4 illustrates the compression of this simple graph with proposed coding scheme. Figure 3.4 (a) shows the adjacency matrix of the graph in Figure 3.3 and Figure 3.4 (b) displays the relation pattern of two merged vertices and the adjacency matrix of remaining vertices. In Figure 3.4, both (a) and (b) convey a model of graph representation, and the coding cost of each model can be calculated using Eq. (3.6). By finding more similar nodes from the adjacency matrix of remaining vertices, we obtain a set of models to represent the graph. Then we use the MDL principle to choose the best one.

3.5.2 The Greedy Merging Algorithm Info-spot

The intuition of the algorithm Info-spot is that if a pair of nodes have very similar link patterns, we can use one vertex's link pattern and the relation pattern between them to represent them. As for any node pairs, we can merge them by removing the vertex with a higher entropy link pattern so as to reduce the coding cost. The defined similarity measure between any node pair is based on the relation pattern between them. Therefore, the more similar the link pattern of two vertices is, the less bits are required to express the relation pattern of these node pair. Typically, in each graph there are many pairs of nodes that can be merged to give a reduction in coding cost. Info-spot iteratively find the most similar pairs in the remaining vertices and merge them by removing the one with higher entropy until all vertices are merged, and meanwhile in each iteration we record the better MDL representation and further get the optimal one.

Now we describe the Info-spot algorithm, which is shown in Algorithm 3.1. Specifically, it can be divided into two phases: Initialization and Iterative merging. In the Initialization phase, we first compute the similarity between all pairs of vertices via their relation pattern, then we set the remaining vertices V_R to be the whole vertices set V_G and different pattern D_P to be empty, after that we compute the initial coding cost $minCost$ with V_R and D_P , which is the entropy of original adjacency matrix. During the Iterative merging phase, we first pick the pair (a, b) with the minimum similarity in link pattern. Then we remove the vertex with higher entropy from V_R and add the relation pattern of (a, b) to D_P . After that we recompute the coding cost with new V_R and D_P , and record the smaller one compared with $minCost$. Then we merge another pair of vertices again until all vertices are merged together. The following example illustrates the greedy algorithm on a simple graph.

Figure 3.5 depicts the process of Info-spot on the example graph. After acquiring the similarity between each node pair, we find the node pair $(1, 2)$ is the most similar one, then they merged and meanwhile the link pattern of node 1 is deleted from the adjacency matrix, so the five vertices adjacency matrix is replaced by the four vertices 2, 3, 4, 5 adjacency matrix and the relation pattern between node pair $(1, 2)$, which is expressed as $R(1, 2)$. The following process can be deduced analogically until only one vertex 5 is left in the adjacency matrix. Finally, the graph

Algorithm 3.1 Info-spot

Input: Graph G with all vertices V_G , Different patterns D_P , Remaining vertices V_R **Output:** Interesting Vertices: R **//Initialization:**

- 1: Set $V_R = V_G$ and $D_P = \emptyset$;
- 2: **for** Each node pair $(a, b) \in V_G$ **do**
- 3: Compute the similarity via relation pattern $R(a, b)$ by Eq. (3.4);
- 4: **end for**
- 5: Calculate the initial coding cost $minCost = CC(V_R, D_P)$ by Eq. (3.6);

//Iterative Merging:

- 6: **while** $V_R \neq \emptyset$ **do**
 - 7: Choose the node pair $(a, b) \in V_R$ with the minimum similarity;
 - 8: Choose the node v from node pair (a, b) with the larger entropy;
 - 9: Remove node v from the V_R ;
 - 10: Add $R(a, b)$ to D_P ;
 - 11: Compute the new coding cost $newCost = CC(V_R, D_P)$;
 - 12: **if** $newCost < minCost$ **then**
 - 13: Set $minCost = newCost$, $V_{R_{best}} = V_R$ and $D_{P_{best}} = D_P$;
 - 14: **end if**
 - 15: **for** Each node $c \in V_R$ connected with v **do**
 - 16: Recompute the similarity of node pair (c, r) , $\forall r \in V_R$;
 - 17: **end for**
 - 18: **end while**
 - 19: Choose isolated nodes $R = V_{R_{best}}$;
 - 20: **return** R ;
-

can be showed in the form of one vertex 5 adjacency matrix and 4 relation patterns. In each step, we record the coding cost for each model, combined with adjacency matrix and relation patterns, and finally choose the best model with MDL principle.

3.5.3 Runtime Complexity

The complexity of the Info-spot algorithm is $O(|V|^2 \cdot d_{av} + |V|)$. Where $|V|$ is the number of vertices, d_{av} is the average degree of each vertex.

3.6 Experiments

To extensively study the performance of our algorithm, we conduct experiments on two excerpts of DBLP dataset¹ and one excerpt of Enron email dataset² with Oddball [2]. We implement our algorithm in Java, and the source code of Oddball is obtained from the author. All experiments have been performed on a laptop with 2.0 GHZ CPU and 4.0 GB RAM.

To facilitate the evaluation and interpretation we focus on the Data Mining research field when extracting experimental data sets from DBLP. Thus, we generate two collaboration graph data sets from DBLP, one is the coauthor network of 5 famous scientists in Data Mining, the other one is the coauthor network in Data Mining field. For Enron email dataset, we extract a communication graph from three ex-CEO of Enron for further comparison.

3.6.1 Coauthor Network of 5 Famous Scientists

Comparison with Oddball. This data set is extracted from DBLP, the vertices are 5 famous scientists (Christos Faloutsos, Jiawei Han, Philip S. Yu, Qiang Yang and Chris H. Q. Ding) and their coauthors in data mining field, if any two of them have published papers together, there is an edge between them, the data set contains 1297 vertices and 7420 edges. We spot interesting nodes of this data set with Info-spot and compare them with the outliers detected by Oddball. Info-spot

¹<http://dblp.uni-trier.de/xml/>

²<http://www.cs.cmu.edu/~enron/>

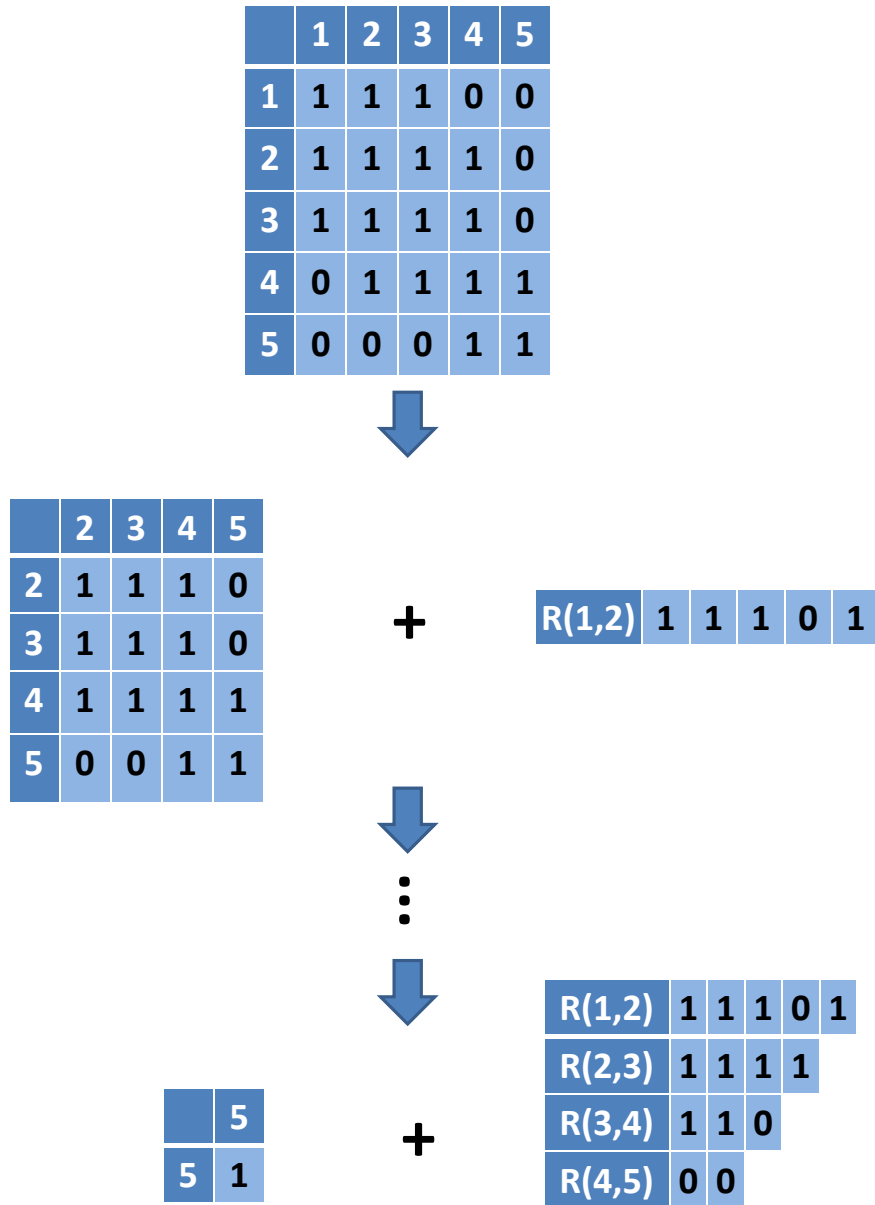


Figure 3.5: The Process of Compression

automatically outputs 14 interesting nodes and we rank them by their information contents. For comparison, we pick the top 14 ranked outliers that Oddball outputs. Table 3.1 summarizes the results highlighting the names of 5 scientists chosen to generate the graph in bold.

It can be clearly seen from the table that Oddball can only detect 4 famous scientist which we chose to generate the graph, it can not find Chris H. Q. Ding. However Info-spot can detect all 5 famous scientists, and all of them are ranked in a higher position than that by Oddball. Furthermore, Info-spot can automatically select 14 outliers without any prior information, in the other hand Oddball only give a rank list, users have to choose a threshold to cut off. Additionally, Oddball detected some guys who are not famous, like Gene Spafford and Jim Massaro who are in the top and third of rank list of Oddball. Oddball treats them as an outliers mainly because they fit the assumption that the number of nodes and the number of edges of a Egonet follow a power law, however ignores the information content: Gene Spafford and Jim Massaro only publish 4 and 1 papers, respectively, which provide little information to users. In the other side, Info-spot provides information-rich nodes, like Jian Pei whose information content is even more than Chris H. Q. Ding whom we choose to generate the dataset. We get academic statistic for Jian Pei from <http://www.arnetminer.org>, the H-index of Jian Pei is 39, being ranked the top-31th scientist in data mining and the citation of him is 10691 being ranked the top 30th in same area, which shows that Jian Pei is a very famous and influential scientist in the field of data mining.

Visualization. To emphasize the effectiveness and strength of the information-based similarity measure, we use Multidimensional Scaling technique to map the similarity matrix to lower dimensions for visualization. The best representation dimensions are chosen by Bayesian Information Criterion which was introduced in Section 3.4. The best dimensions to represent the similarity matrix of 5 scientist dataset are 4, Figure 3.6 depicts the plot in each two mapping dimensions of the similarity matrix of 5 scientists dataset. We provide some examples detected by Info-spot in the figure. This experiment clearly shows that in the embedding the interesting nodes which contain much more information are far from the ordinary nodes. Thus, they can also be detected by established feature space outlier detection techniques e.g. LOF [12].

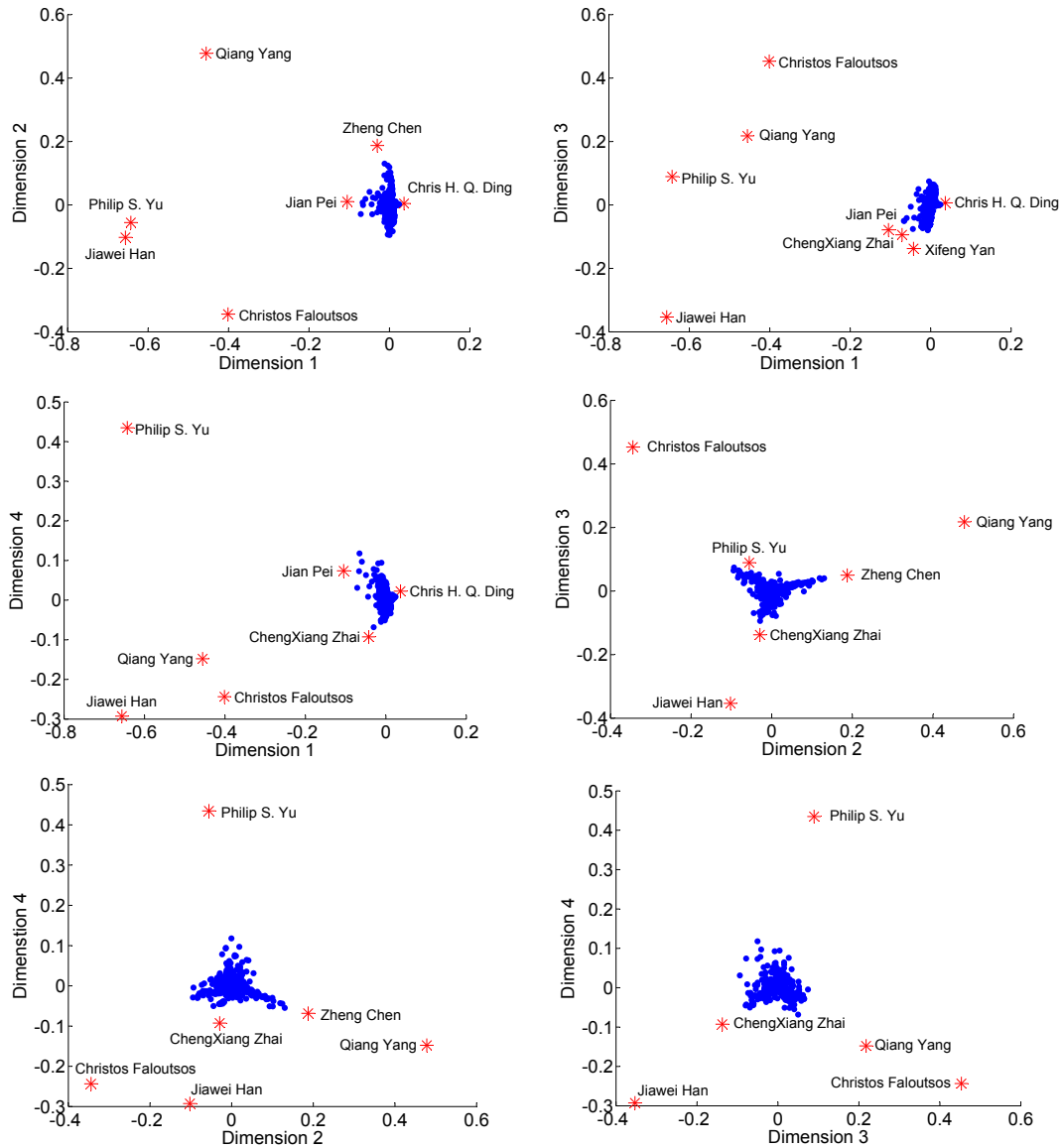


Figure 3.6: Mapping Similarity Matrix of 5 Scientists Data Set to 4 Dimension and Plot in Each Two Pairs (Most interesting nodes are highlighted with their name).

Table 3.1: Results of Info-spot and Oddball on 5 Famous Scientists Dataset

Rank	Info-spot	Oddball
1	Jiawei Han	Gene Spafford
2	Philip S. Yu	Jiawei Han
3	Christos Faloutsos	Jim Massaro
4	Qiang Yang	Philip S. Yu
5	Jian Pei	Alain Bensoussan
6	Chris H. Q. Ding	Qiang Yang
7	Jeffrey Xu Yu	Nathan Berg
8	Zheng Chen	Christos Faloutsos
9	Haixun Wang	Joel Sachs
10	ChengXiang Zhai	Alex Y. M. Chan
11	Xifeng Yan	Ravi S. Sandhu
12	Bing Liu	Shari Roseman
13	Wei Fan	Shouhuai Xu
14	HongJiang Zhang	Su-Jeong Ko

3.6.2 Data Mining Scientists Collaboration

This data mining scientists collaboration dataset is extracted from the DBLP, we choose the authors who published at least one paper in KDD or ICDM to be the vertices of graph, if two authors have collaboration in any conference or journal, there is an edge between them. This dataset consists of 6416 nodes and 26457 edges.

Proposed Info-spot outputs 16 interesting nodes finally, Table 3.2 provides the top 16 outstanding nodes detected by Info-spot and Oddball, highlighted in bold are all scientists that are identified by both algorithms, for Oddball we pick the top 16 in the rank list. From the table we can clearly show that very famous scientists in data mining have been selected by both algorithms, like Jiawei Han, Philip S. Yu, Christos Faloutsos and so on, however some other famous scientists like Jeffery Xu Yu, Tao Li and Heikki Mannila are missed by Oddball.

3.6.3 Email Communication Network of 3 ex-CEO from Enron

This email communication graph data is extracted from Enron email network, the nodes are three ex-CEO's email address of company Enron(Jeffery Skilling, John Lavorato and Kenneth

Table 3.2: Results of Info-spot and Oddball on Data Mining Dataset

Rank	proposed	Oddball
1	Jiawei Han	Pelin Angin
2	Philip S. Yu	Jiawei Han
3	Christos Faloutsos	Philip S. Yu
4	Qiang Yang	Yan Chen
5	Jian Pei	Chen Xi
6	Wei-Ying Ma	Qiang Yang
7	Li Zhang	Christos Faloutsos
8	Wei Wang	John M. Rosenberg
9	Jeffrey Xu Yu	Jian Pei
10	Xindong Wu	Wensi Xi
11	Bing Liu	Arun K. Majumdar
12	Tao Li	Yi Liu
13	Zheng Chen	Wei-Ying Ma
14	Heikki Mannila	Benyu Zhang
15	Ke Wang	Zheng Chen
16	Rakesh Agrawal	Horia-Nicolai L. Teodorescu

Lay) and their colleagues' which has communicated with them. If one person sent an email to the other one, there is an edge between them. The dataset contains 4177 nodes and 19281 edges.

Proposed Info-spot automatically outputs 18 interesting nodes finally, for Oddball we pick the top 18 in the rank list. Table 3.3 provides the top 18 outstanding nodes detected by Info-spot and Oddball, in which the postfix '@enron.com' is omitted. All email addresses that are identified by both algorithms are highlighted in bold, the email addresses of 3 ex-CEO that we used to extract the graph are with asterisk. From the table we can clearly see that some interesting nodes can be detected by both algorithms, but surprisingly Oddball can not find 3 ex-CEO's email addresses, which we used to generate the graph. Info-spot can find them and some other interesting nodes that Oddball missed, like 'lavorato' is another email of CEO John Lavorato, and 'louise.kitchen' is the ex-president of Enron online department.

Table 3.3: Results of Info-spot and Oddball on Email Communication Dataset

Rank	Info-spot	Oddball
1	kenneth.lay*	bob.ambrocik
2	jeff.skilling*	rodolfo.acevedo
3	bob.ambrocik	bodyshop
4	john.lavorato*	steve.woods
5	rosalee.fleming	veronica.espinoza
6	sherri.sera	jeff.royed
7	veronica.espinoza	amelia.alder
8	technology.enron	forster
9	lavorato	technology.enron
10	bodyshop	enron.expertfinder
11	billy.lemmons	jordan
12	joannie.williamson	mbkuntz
13	amelia.alder	zipper
14	cliff.baxter	henry.safety
15	louise.kitchen	lawrence.ciscon
16	karen.denne	leigh.edwards
17	zipper	jim.fallon
18	outlook.team	whalen.kevin

3.7 Related Work and Discussion

Outlier detection is one of the most important data mining task. For vector data, many mature algorithms have been proposed to detect outstanding objects. These methods belong to three categories: parametric statistical methods, distance-based and density-based techniques. Statistical parametric methods assume that all objects follow some underlying distributions except the outliers [46]. Distance-based methods define an outlier as it is d distance far away from the other n points [59]. Density-based methods are derived from density-based clustering, and it detects outliers which do not fit well into its neighborhood density, e.g. LOF [12], LOCI [87], CoCo [9]. All these approaches have in common that they search for data points which are far away from the rest of data set. However, in terms of graph data, the notion outlier should be redefined.

The structure of the graph data is various and more challenging for outlier detection. In graph data, the definition of outlier is vitally interrelated with its link structure. Besides in different application field, owing to the addition of attribute, the definition of outlier is even more versatile.

Therefore, it is impossible to transfer the notion and methods of vector data directly on graph data.

As well as Info-spot, some proposed outlier detection algorithms for graph data are based on MDL principle. Eberle and Holder [27] and Nobel and Cook [84] consider specific unusual substructure which is missing or attaching from the normal substructure as anomalous in graph, both of their anomalies detection algorithm based on the SUBDUE which is a algorithm for finding repetitive pattern in graph. Chakrabarti [15] detect anomalous edge by measuring the cost of removal of this edge by MDL. In contrast with them, Info-spot explicitly focus on nodes but not subgraphs or edges, which is totally different.

Some other methods focus on outstanding nodes detection as well, such as Lin and Chalupsky [22] proposed an algorithm in discovering unusual interestingly linked entities in multi-relational data. Moonesinghe and Tan [78] use Markov chain model to detect the outliers in graph and compute the outlier score. For the bipartite graphs, according to compute the neighborhood information for each node by random walk, Sun et al. [104] identify outstanding nodes. Recently, an interesting approach was proposed by Akoglu et al. [2]. They consider the link pattern of node and its neighbors which is in the form of near-cliques and near-star as interesting. Four features are extracted to represent the nodes of a graph in a low-dimensional vector space. Interesting near-cliques and star-like structures can be identified with standard outlier detection techniques for vector data. However, the feature extraction technique will lose information by transforming the graph data to vector data. Compared with these methods, we give a new definition of outlier in graph and propose Info-spot, which automatically flags the most interesting nodes of a graph without requiring any input parameters.

3.8 Chapter Conclusion

In this chapter, we focused on automatically detecting the most interesting nodes in a large graph. We addressed this challenge from the information-theoretic perspective of data compression which has attractive advantages:

We consider the link pattern of each node as the only source of information for knowledge

discovery. This information is directly available in the adjacency matrix of each graph and our approach does not require any further assumptions, models or input parameters. The description length or information content in bits of a link pattern is an intuitive indicator for the interestingness of a node. However, to comprehensively assess the interestingness of a node, we need consider its information content in the context of the other nodes.

Therefore, we proposed Info-spot, an efficient greedy algorithm for automatically spotting all interesting nodes. Info-spot iteratively merges those pairs of nodes having the most similar link patterns. Controlled by the objective of data compression, the merging process continues until only nodes with an exceptional information-rich link pattern remain isolated. In comparison to most existing approaches for outlier detection in graphs, Info-spot has two major benefits: Our approach automatically flags outstanding nodes without requiring any parameters, assumptions or thresholds. Furthermore, by an information-theoretic view on the link pattern combined with the greedy merging algorithm, Info-spot naturally and flexibly defines what is an interesting node in that particular graph data set under consideration. In ongoing and future work, we want to focus on a comprehensive information-theoretic framework for graph summarization, clustering and outlier detection.

Chapter 4

Compression-based Graph Mining

Exploiting Structure Primitives

In this chapter, we continue discussing about knowledge discovery in the simple graph. Instead of nodes, we focus on studying structures of a simple graph. Graph is a structural data, thus analyzing structures is significant in graph mining. One goal of this chapter is to achieve new kinds of clusters from graph. As illustrated in Chapter 2, traditional graph mining tasks, especially with regard to graph partitioning, focus on discovering dense patterns from graph data. However, most graph data generated from the real world data are very sparse or some parts of them are sparse. It is difficult to detect dense patterns in such graphs. Therefore, in this chapter we analyze the graph from the viewpoint of structures and try to discover novel type of clusters. The other goal is to handle multiple graph mining tasks simultaneously. We are able to detect novel clusters, distinguish graphs and predict links in one algorithm. A novel algorithm CXprime (Compression-based eXploiting **P**rimitives) is proposed. The core part of CXprime is the proposed novel coding schemes by using the three-nodes substructures. With the new coding schemes, we are able to form both triangle- and star-like structures. Based on the Minimum Description Length (MDL) Principle, graph is effectively compressed by using the three-nodes substructures. Meanwhile, we can distinguish a graph by its structure, e.g. triangle structures or star structures. In addition, novel star structure clusters can be detected in a graph, which

can not be found from classical methods. Additionally a novel structure based unsupervised link prediction method is proposed. Guided by MDL principle, the algorithm fulfills all the graph mining tasks without any input parameters.

The remainder of this chapter is organized as follows: Section 4.1 introduces the motivations and contributions of the work in this chapter. Section 4.2 describes the novel coding scheme in more detail. Section 4.3 presents the proposed algorithm. Section 4.4 shows the experiments and evaluations. Section 4.5 discusses related work and Section 4.6 concludes the chapter.

Parts of the material presented in this chapter have been published in [30], where Jing Feng was responsible for the development of the main concept, implemented the main algorithms and wrote the largest parts of the paper; Xiao He helped with the development of the main concept, performed parts of experiments and wrote part of the paper; Nina Hubig wrote part of the paper and revised the whole paper; Christian Böhm and Claudia Plant supervised the project and made contributions to the building of coding scheme and revised the whole paper. The co-authors also contributed to the conceptual development and paper writing.

“Jing Feng, Xiao He, Nina Hubig, Christian Böhm, Claudia Plant. Compression-Based Graph Mining Exploiting Structure Primitives. The IEEE International Conference on Data Mining (ICDM) 2013: 181-190.”

4.1 Introduction

4.1.1 Motivation

Real world data from various application domains, such as social networks, bioinformatics and neuronal networks can be modeled as graphs. Specific topological structures like triangles and stars represent meaningful characteristic relationships among subsets of nodes. Specifically, in [94] the authors introduced a transitivity attribute which is calculated from the fraction of triangles in all node triplets. As an indispensable condition for small-world networks, high transitivity implies more triangles in a graph. Moreover, the authors in [54] characterize graphs with a power-law degree distribution, which is a significant feature of scale-free networks, by a very

low degree of most vertices combined with a high degree of only very few vertices. Therefore, hubness plays a pivot role in a graph which is created in star style. Obviously, triangles and stars are the two basic regular substructures which appear in graphs most frequently. The triangle embodies the transitivity of a graph and the star shows the hubness of a graph. Moreover, graphs containing more triangles are showing different structures and characteristics to graphs that contain more stars. It is very interesting to know which substructure is popular in a graph. Firstly, the popular substructure reflects the structure feature of the whole graph. Secondly, based on the frequent appearance of a substructure, the graph can be compressed under Minimum Description Length (MDL) principle [92]. Thirdly, the structure information is very helpful for link prediction.

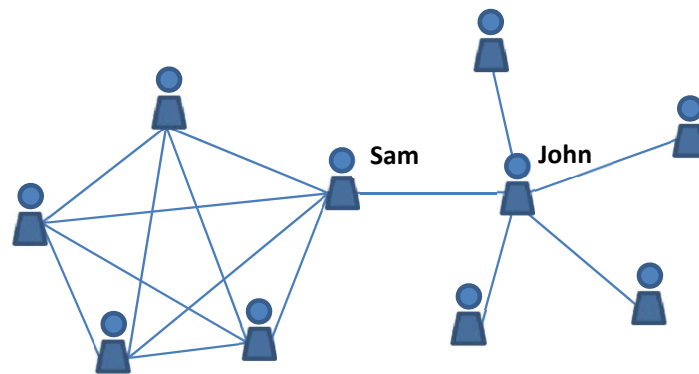


Figure 4.1: Two Differently Structured Sub-graphs.

However, there are some graphs that possess both high transitivity and hubness in different parts of the whole graph. Take the toy graph which is shown in Figure 4.1 as an example. The graph displays the friendship relationship between Sam and John. Supposing that Sam prefers to make friends with local people, and all his local friends are also friends with each other; while John's friends scatters in various countries, so that some of them do not know each other. Therefore, the circle of friends of Sam performs a clique with a large amount of triangles while John is the hub of his circle of friends which is displaying a star. The authors of [85] point out that transitivity increases with the strength of the corresponding community. Traditionally, many algorithms are designed to detect communities which pursue the compactness of inner vertices and sparseness of intra vertices, including spectral clustering [99], Min-Max cuts [26] and others.

However, as shown in Figure 4.1, it is meaningful to distinguish among the star-like cluster as John's circle of friends and the clique-like cluster of Sam's friends.

4.1.2 Contributions

In this chapter, we propose a novel compression-based graph mining algorithm named CXprime (Compression-based eXploiting **P**rimitives). The algorithm is based on exploiting three-node primitives which are the smallest substructures and express both transitivity and hubness of a graph. Unlike complex substructures, three-node primitives are simple and easy to count. Any graph no matter how complex it is can be considered as a combination of three-node primitives. Moreover, we exploit the differences in the relative frequency of three-node primitives for graph compression. Based on the idea of MDL, frequently appearing primitives are effectively compressed in short bitstrings and rarely appearing primitives represented by longer bitstring. Due to the fact that three-node primitives are appropriate for representing both triangular and star substructures, CXprime is designed to distinguish and partition graphs with different substructures. The main contributions of CXprime are summarized as follows:

- **Discovery of graph structure:** CXprime automatically discovers the basic structure type of a graph. Relating data mining to data compression, the core of CXprime is a coding scheme based on three-node primitives allowing to model both triangular and star structures. Emphasizing the characteristics of a triangle graph and star graph separately, CXprime comprises a Triangle Coding Scheme and a Star Coding Scheme. By applying these two coding schemes to an unknown graph we can determine its structure type either as star-like or triangle-like.
- **Graph partitioning based on structures:** Based on the Triangle Coding Scheme and Star Coding Scheme which is proposed by CXprime, the graph can be partitioned into subgraphs with star or triangular structures. Furthermore, the number of clusters can be selected automatically since CXprime is based on the idea of Minimum Description Length principle.

- **Link prediction based on structures:** CXprime allows us to detect the structure type of a graph. We exploit this information to design a novel unsupervised link prediction score. Experiments demonstrate that this structure-based score outperforms existing techniques which impressively demonstrates that structure information is very useful for graph mining.

4.2 Graph Compression using Structure Primitives

Suppose we want to transfer a graph G over a communication channel from a sender to a receiver. We consider an unweighted and undirected graph G with $|V|$ nodes in this chapter. The graph is provided by its adjacency matrix \mathcal{A} with entries $\mathcal{A}_{i,j}$ specifying whether there is an edge among the nodes i and j . To transfer G we need to compress each entry in its adjacency matrix. If we do not have any knowledge on the structure of G , the coding costs are provided by the entropy of the adjacency matrix. Since G is an undirected graph, \mathcal{A} is symmetric and we only need to encode the upper or lower half of the matrix without the diagonal (representing self-links which are never or always set by convention). Regardless of the type of graph and its characteristics, which can be e.g. scale-free, small-world, Erdos-Renyi, clustered, dense or sparse, we can represent every graph by a bit string of length corresponding to the entropy of its adjacency matrix \mathcal{A} . To encode a single entry $\mathcal{A}_{i,j}$, we need an average of $H(\mathcal{A})$ bits, where $H(\mathcal{A})$ denotes the entropy and is provided by:

$$-(p(e) \cdot \log_2(p(e)) + p(ne) \cdot \log_2(p(ne))),$$

where $p(e)$ stands for the probability to observe an edge in G , corresponding to the percentage of 1s in \mathcal{A} , and $p(ne)$ analogously. Thus the total coding costs are provided by: $|V| \cdot (|V| - 1)/2 \cdot H(\mathcal{A})$, where $|V|$ is the number of vertices.

If G contains regularities in the form of frequent structure primitives like triangular or star structures, we can compress it more effectively than its entropy. Note that our primary focus is not on compacting the data for transmission over a communication channel but on mining the truly relevant patterns in the data in an unsupervised way. However, there is a direct re-

relationship between data compression and knowledge discovery: The better a set of patterns fit to the data, the better is the compression, i.e. the greater are the savings in coding costs over the entropy which serves as a baseline. In the following section, we elaborate concrete coding schemes including structure primitives which are characteristic for major types of real world graphs, including small-world and scale-free.

4.2.1 Basic Coding Paradigm

Three-node primitives (substructure with three nodes) are the smallest substructures which can embody both connectivity and transitivity of a complex graph. Figure 4.2 enumerates all possible link patterns among three nodes in an undirected graph. In a random graph, each edge exists with the same likelihood and does not depend on the existence of other edges. Therefore, a random graph requires coding costs corresponding to its entropy and cannot be represented more efficiently. If a graph is characterized by transitivity or star-like structures, the existence likelihood of an edge depends on the existence of other edges. In a graph with many star-like hubs, if e.g. node B is already connected to node A (cf. Figure 4.2 (c)), then node C is also connected to A with a high likelihood (cf. Figure 4.2 (g)). In a highly transitive graph, when we know that there are two edges among three nodes (cf. Figure 4.2 (g)), also the third edge closing the triangle (cf. Figure 4.2 (h)) exists with a high likelihood. In other words, the probability of observing the third edge BC (cf. Figure 4.2 (h)) is very high under the condition that we already observed a potential triangle formed by two edges. We can exploit these typical variations in conditional probabilities to effectively compress structured graphs as follows (see Figure 4.3):

Fixed Processing Order for Coding and De-Coding. First, the sender and the receiver agree on some fixed order for encoding and de-coding the adjacency matrix \mathcal{A} , which can be column-wise, row-wise or diagonal-wise. For an undirected graph, the code is a bitstring composed of $|V| \cdot (|V| - 1)/2$ codewords. By the fixed coding and de-coding order, the receiver always knows which codeword corresponds to which entry $\mathcal{A}_{i,j}$ without any ambiguity or information loss. To encode the graph in Figure 4.3(a), without loss of generality, we select a diagonal-wise order of processing. The diagonal colored in black in Figure 4.3(b) does not need to be transferred, since

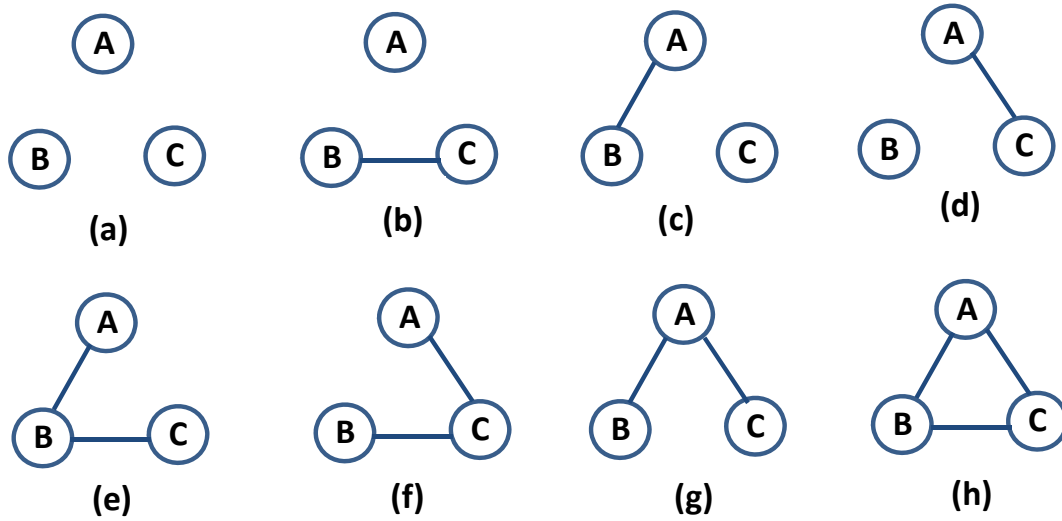


Figure 4.2: All Possible Connections of Three-node Primitives

it represents self-connections which are set by convention. To encode the first off-diagonal, we cannot use any conditional probabilities since each entry corresponds to a single edge among two different nodes and we have no information on three-node primitives.

Case Distinctions based on Conditional Probabilities. Starting from the second off-diagonal, we can exploit conditional probabilities together with case distinctions depending on the information we have already seen before in the processing order. In particular, we define three conditional probabilities which can be obtained from counting the relative frequency of three-node primitives (cf. Figure 4.2), e.g. N_a is the frequency of case (a) :

Definition 4.1 (Basic Conditional Probabilities) *Basic conditional probabilities contain No Edge, Potential Star and Potential Triangle:*

- **No Edge:** We have not seen any edges in three-node primitives so far.

$$p(e|\mathbf{No\ Edge}) = \frac{N_{bcd}}{N_a + N_{bcd}}$$

- **Potential Star:** We have already seen one edge. If another link were added next, we would get a star primitive, therefore we call this primitive with only one edge potential star.

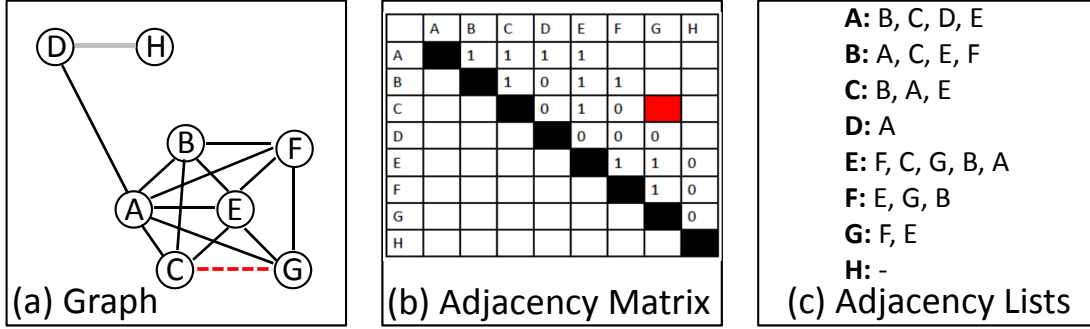


Figure 4.3: Example for Coding and De-coding. (a) Graph; black edges: already coded; red edge: current edge; grey edges: not yet processed; (b) Adjacency matrix: filled entries: already coded in diagonal-wise order of processing; red: current entry; (c) Current stage of adjacency lists, see Section 4.2.1.

$$p(e|\mathbf{Potential Star}) = \frac{N_{efg}}{N_{bcd} + N_{efg}}$$

- **Potential Triangle:** We already observed two edges. If another link were added next, we would get a triangle primitive, therefore we call this primitive with two edges potential triangle.

$$p(e|\mathbf{Potential Triangle}) = \frac{N_h}{N_h + N_{efg}}$$

Where $N_{bcd} = (N_b + N_c + N_d)/3$ and $N_{efg} = (N_e + N_f + N_g)/3$, cf. Figure 4.2. For every condition $\sigma \in \{\mathbf{No Edge}, \mathbf{Potential Star}, \mathbf{Potential Triangle}\}$, the probability that no edge exists is provided by $p(ne|\mathbf{Condition } \sigma) = 1 - p(e|\mathbf{Condition } \sigma)$. The entropy of each condition is provided by:

$$H(\sigma) = -(p(e|\sigma) \cdot \log_2 p(e|\sigma) + p(ne|\sigma) \cdot \log_2 p(ne|\sigma)). \quad (4.1)$$

where e means there is an edge and ne for no edge.

When encoding a particular entry in the adjacency matrix, we can find all previously seen three-node primitives by inspecting the nodes in the corresponding row and column. In our example (cf. Figure 4.3), we are currently coding the third off-diagonal and now want to encode the red entry corresponding to the edge CG in row 3 and column 7. We already have information

about the nodes D, E, and F. We know that D is not connected to C nor G from the entries (3, 4) and (4, 7). We also know that E is connected to C and G ((3, 5) and (5,7)). Finally, we know that F is connected to G but not to C ((6,7) and (3,6)). We observed all three conditions above in the previously seen data. Which one should we use to encode the current entry? Since we aim at compressing the data as much as possible we always select that condition which is expected to give us the shortest codeword. This is the condition having the lowest entropy. We could also see our three conditions as alternative classifiers predicting the current entry. If we have the choice among several classifiers, i.e. if multiple conditions apply, it makes sense to select the classifier which is most certain about the current case. For our example graph we have $p(e|\mathbf{Potential\ Triangle}) = 0.72$, $p(e|\mathbf{Potential\ Star}) = 0.35$ and $p(e|\mathbf{No\ Edge}) = 0.68$. Thus, we select the condition **Potential Triangle**, since it allows us to encode the current entry with 0.86 bits in average, while **Potential Star** would require 0.93 bits and **No Edge** 0.9 bits. The current entry is an edge, so we use the codeword representing an edge under the condition **Potential Triangle** to encode it. This coding scheme can be decoded without information loss since for coding and decoding sender and receiver perform the same case distinction based on the same data.

Parameter Costs. To decode the bitstring, the sender needs the codebook consisting of the conditional probabilities required to perform the case distinction and the code table saying which bitstring represents an edge or no edge in every case. Following [91], these parameter costs can be approximated by Eq. (4.2):

$$CC_{param} = 0.5 \cdot num \cdot \log_2 \frac{|V| \cdot (|V| - 1)}{2}, \quad (4.2)$$

where $|V|$ is the number of nodes in the graph and num denotes the number of parameters, which is three in our case since we consider three different conditional probabilities.

Efficient Implementation with Adjacency Lists. For efficient coding and de-coding the sender and receiver use adjacency lists. Every time a new entry is processed, it is inserted into the adjacency lists of both corresponding nodes. Figure 4.3(c) displays a snapshot of the adjacency lists before processing CG. In order to collect the applicable cases for encoding or decoding CG,

instead of looking into the corresponding row and column of the adjacency matrix, we scan the adjacency lists of nodes C and G. In particular, we start with the list of G from the beginning and with that of C from the end. In the first step, we retrieve F as the first node in the list of G and E as the last node in the list of C. We know that F is adjacent to G but not to E, which means that the condition **Potential Star** is applicable. Having processed F, we move one step forwards in the list of G and detect the matching node E, from which we can deduce that **Potential Triangle** also is applicable. Having processed E, we can move one step in both lists, which means that we come to the end of the list of G and obtain B in the list of C. Due to the processing order, information on three-node primitives formed with node B is not yet available, therefore we can stop as soon as we detect a node coming before C. But we know that we already have information on D. Therefore, we detect that condition **No Edge** is also applicable.

4.2.2 Extended Coding Paradigm

The basic coding paradigm only considers the primitives with three nodes, which is the simplest substructure in a graph. During the diagonal-wise coding process, we can see less previous information in the beginning and more at the end. Therefore if primitives with more nodes can be used, we could compress the graph more effectively and get more knowledge from it. However, counting probabilities of primitives with more nodes would require high computational costs.

We choose some primitives with more nodes to extend the basic coding paradigm as Figure 4.4 shows. These primitives are frequent in real-world graphs, like dense communities or hub nodes and their neighbors.

Definition 4.2 (Higher-order Conditional Probabilities) *Higher-order conditional probabilities contain Multiple Triangles and Strong Star:*

- **Multiple Triangles:** *There are multiple points connecting both B and C, which means previously we can see multiple dual-connected edges. As shown in Figure 4.4 (a) and (b);*
- **Strong Star:** *There are multiple points connecting B or C, which means previously we can see multiple single-connected edges. As shown in Figure 4.4 (c) and (d).*

Suppose the high-order conditional are represented as **Condition** $\sigma' \in \{\text{Multiple Triangles}, \text{Strong Star}\}$. The probabilities of existence an edge e under the condition σ' are provided by:

$$p(e|k \cdot \text{Condition } \sigma') = \frac{\sum_{i=1}^{|E|} C_{m_i}^k}{\sum_{i=1}^{|E|} C_{m_i}^k + \sum_{j=1}^{|NE|} C_{n_j}^k}, \quad (4.3)$$

Where $|E|$ is the number of edges in a graph G , and $|NE|$ is the number of pairs of nodes without connection. m_i with $\{i = 1, 2, \dots, E\}$ is frequencies of **Condition** σ' for the connected entry, and n_j with $\{j = 1, 2, \dots, NE\}$ is frequencies of **Condition** σ' for the unconnected entry. C is combination symbol. k is the actual frequency we can see for each entry.

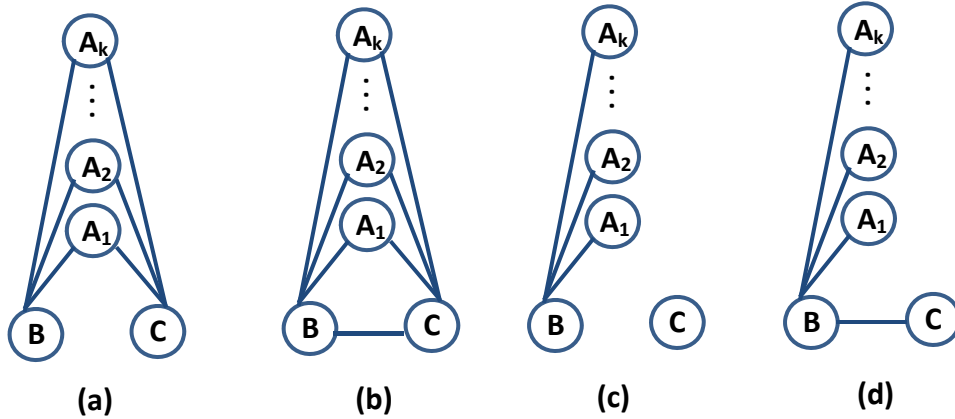


Figure 4.4: Multiple Primitives.

No extra effort is required to obtain these higher-order conditional probabilities, since they are calculated without counting the number of edges (again), but are created by calculating the combination of pair of nodes that have the same basic primitives, e.g. two linked nodes share n neighbors then they have C_n^k triangles. We use Eq. (4.3) to calculate the statistic for multiple triangles and strong stars. In our experiments of both synthetic and real data sets, the probabilities in Eq. (4.3) with different k become stable when k increases. Therefore, we only compute the probabilities when $k \leq 4$.

To be able to automatically detect the structure type of a graph we now introduce a *Star Coding Scheme* and a *Triangle Coding Scheme*.

Definition 4.3 (Star and Triangle Coding Scheme) *The Star Coding Scheme employs for coding the following set \mathcal{S} of conditional probabilities $\mathcal{S} := \{\text{No Edge, Potential Triangle, Potential Star, Strong Star}\}$ while the Triangle Coding Scheme works with a different set $\mathcal{T} := \{\text{No Edge, Potential Star, Potential Triangle, Multiple Triangles}\}$ as specified in the basic and higher-order conditional probabilities, cf. Definitions 4.1 and 4.2. Upon encoding or decoding each entry, the sender or receiver always select that conditional probability of \mathcal{S} and \mathcal{T} having the lowest entropy respectively. The overall coding cost CC_S for the Star Coding Scheme is provided by:*

$$CC_S = \sum_e \frac{L(e|\mathcal{S})}{\min H(\mathcal{S})} + \sum_{ne} \frac{L(ne|\mathcal{S})}{\min H(\mathcal{S})} + CC_{param} \quad (4.4)$$

where $L(e|\mathcal{S}) = -\log_2 p(e|\mathcal{S})$ is the coding length of a connected entry, $L(ne|\mathcal{S}) = -\log_2 p(ne|\mathcal{S})$ is the coding length of a unconnected entry, CC_{param} represents the parameter costs as specified in Eq. (4.2). However, we now have $3 + k - 1$ parameters to consider. Three of them are basic probabilities, while $k - 1$ of them are higher order probabilities except for corresponding basic three-node primitive. The coding cost CC_T of the Triangle Coding Scheme is determined analogously applying the corresponding set \mathcal{T} .

Both coding schemes contain all basic conditional probabilities since we need to be able to represent any possible link pattern among three-node primitives with both schemes. The higher-order probabilities emphasize and reward star and triangle structures by assigning very short bitstrings to them. The coding and decoding process with these two extended coding schemes works as explained in Section 4.2.1. Also here, to encode the first diagonal of \mathcal{A} , we use the entropy provided by the general edge existence probability, which is omitted in the above definition for clarity of presentation.

4.3 Algorithm CXprime

In this section, we present our algorithm CXprime to mine useful information considering the underlying structure primitives from graphs. The core part of CXprime are the two coding schemes, Triangle and Star Coding Scheme, which are proposed in the previous section. CXprime is able

to distinguish graphs with high transitivity from graphs with a high amount of hubness by comparing the coding cost of both coding schemes. Apart from distinguishing the graph structures, we combine the coding scheme with K-means for graph partitioning and propose a new link prediction technique exploiting graph structure information.

4.3.1 Graph Partitioning

Considering the complexity of a graph structure, it can be partly in a high transitivity state and partly consisting of many hubs. Traditionally, dense communities with high transitivity are formed by several triangular structures. However, communities with hub are playing a pivot role in graphs as well. Combining the two proposed coding schemes regarding the different structures in a clustering process we introduce a novel idea for graph clustering: partitioning by structure. The K-means like clustering of CXprime is guided by the proposed coding schemes, which partitions graphs into parts based on their respective structural properties that compress the whole graph best. To avoid overfitting, we follow the principle of Minimum Description Length and require that not only the data but also the structure primitives considered in the codebook need to be encoded.

Based on MDL, we extend the proposed Triangle and Star Coding Scheme to be used in clustering. More specifically, the MDL principle is applied to compress a set of candidate clustering models, where in our case different models correspond to different partitions. We use our Triangle and Star Coding Scheme to compress the clusters to test whether the subgraph of an original graph contains triangular or star structures. The coding cost for graph G under the clustering model M with K clusters $\{C_1, C_2, \dots, C_K\}$ is provided by :

$$L(G|M) = \sum_{i=1}^K (\min(CC_S(C_i), CC_T(C_i))) + CC_B \quad (4.5)$$

where CC_S represents the coding cost of the star coding scheme, CC_T is the coding cost of the triangle coding scheme and CC_B indicates the costs describing the edges between different graph clusters, for which simple entropy coding is used.

To avoid overfitting, MDL not only includes the cost for coding the data with the model $L(G|M)$ but also the cost $L(M)$ for the model. We need to specify the clustering assignment and conditional probabilities of the Triangle or Star Coding Schemes for each cluster.

$$L(M) = \sum_{i=1}^K |C_i| \log_2 \left(\frac{|V|}{|C_i|} \right) + \sum_{i=1}^K CC_{param}(C_i) \quad (4.6)$$

where the first term represents the coding cost for the clustering assignment and the second term represents parameters cost.

Finally, the total coding cost for the whole data set with the clustering model M can be obtained by:

$$L(G, M) = L(M) + L(G|M) \quad (4.7)$$

The MDL based clustering algorithm is depicted in Algorithm 4.1. During initialization, we choose K nodes with the longest shortest path between each other as cluster centers. Then neighbors of the center nodes are directly assigned to corresponding clusters. If remaining nodes are neighbors of nodes in a cluster, they will be assigned to this cluster as well. Finally, all nodes are roughly assigned to K clusters and we calculate the star coding cost CC_S and triangle coding cost CC_T of these clusters, then choose the minimum value as initial coding cost. In the iteration phase, each node is moved to all the other clusters to test whether it reduces the coding cost. If the new coding cost is decreased, the new graph clusters will be kept. Otherwise, the nodes will be moved back to their former cluster. The iteration terminates when the clustering labels do not change. Due to the fact that CXprime is a MDL-based algorithm, the number of clusters K can be chosen automatically by searching the minimum coding cost without using any parameter.

4.3.2 Link Prediction

In highly transitive graphs as dense communities, triangular substructures appear the most frequently. It implies that if two nodes are involved in more **Potential Triangle** with other nodes, there will be a higher probability that the two nodes will be linked. On the other hand, star substructures are the most common patterns appearing in a graph with several high degree hubs.

Algorithm 4.1 Graph Partitioning**Input:** Graph G **Output:** Graph Clusters $G_c = C_1, C_2, \dots, C_K$

- 1: Select K nodes as cluster centers, and assign nodes to K clusters;
- 2: Calculate coding cost CC_{old} by Eq. (4.7);
- 3: **while** Converge **do**
- 4: Reassign nodes to other clusters;
- 5: Recalculate coding cost by Eq. (4.7);
- 6: **if** $CC_{new} > CC_{old}$ **then**
- 7: Move nodes back;
- 8: **end if**
- 9: **end while**
- 10: **return** G_c .

Thus if two nodes are involved in more **Potential Star** with other nodes, there is a higher probability that the two nodes will be connected in a star-like graph. Benefiting from structures of graph detected by CXprime, we propose a new unsupervised link prediction method. Specifically, we combine the two situations and give a new prediction score which is shown as:

$$S_{CXprime}(e) = \frac{CC_T}{CC_T + CC_S} \cdot f_T(e) + \frac{CC_S}{CC_T + CC_S} \cdot f_S(e), \quad (4.8)$$

where CC_T and CC_S are the coding cost of Triangle Coding Scheme and Star Coding Scheme respectively, e is the edge that will be predicted, f_T is the frequency of **Potential Triangle** after normalization and f_S is the number of **Potential Star** after normalization. These frequencies can be used for link prediction after we give weights to them based on the graph type. Coding costs of a given graph on both triangle and star type are adopted to generate weights. If $CC_T > CC_S$, then the graph contains more triangular structures than star structures, and the frequency of triangle f_T will be assigned bigger weight. If $CC_T < CC_S$, star structures dominate the graph, thus we give f_S a bigger weight.

Runtime complexity. The runtime complexity of CXprime to compress a graph G with $|V|$ nodes and $|E|$ edges involves: calculating the statistics of structure primitives and coding the adjacency matrix. Gathering the statistics with the adjacency list we need to go through each pair of vertices $O(|V| \cdot (|V| - 1)/2)$ and compare their neighbors $O(2|E|/|V|)$, where $|E|/|V|$ is the average number of edges for each vertex. The asymptotic complexity for this task hence reduce

to $O(|V| \cdot |E|)$. Similarly, the complexity of the coding part is $O(|V|^3)$. For graph partitioning, we use an efficient K-means-style approach which is linear in $|V|$ and usually converges very fast.

4.4 Experimental Evaluation

4.4.1 Discovering the Graph Structure

With the two different coding schemes for stars and triangles, CXprime is able to identify whether the graph is formed by star structures or by triangular structures. It holds that the coding scheme that shows the minimal coding cost for a given graph indicates which structure appearing more frequently in it. We evaluated the two coding schemes of the algorithm on both synthetic data sets and real data sets, the real data sets are coming from sports and media industry. To prove the efficiency of the compression, the acquired coding cost is compared with the entropy of the graph.

Synthetic Data Sets. We generate two types of graphs which clearly show the differences between star and triangular structures. One type is mainly constructed by star-like structure primitives that we call it *star graph*. The other graph is mainly composed of triangular structure primitives which is named *triangle graph*. The number of nodes in both cases is fixed to 100. Specifically, we generate a star-like graph with three hubs, based on which three single star structures with equally same number of nodes are formed. Under the condition of keeping the original star-like structure, noise edges are added to generated graph. The percentage of noise increases from 0.05 to 0.25. Obviously, a clique contains numerous triangular structures. We generate triangle graph based on one clique structure. Similarly, in order to keep the original triangular structure, we remove edges from the generated graph with the percentage of removed edges increasing from 0.05 to 0.25.

As shown in the bar charts of Figure 4.5(a) and 4.5(b), our two coding schemes of CXprime are evaluated on each generated star graph and triangle graph with the percentage of disturbing edges increasing from 0.05 to 0.25 separately. While the entropy of the graph serves as a

baseline. Figure 4.5(a) illustrates that the Star Coding Scheme has a lower coding cost than the Triangle Coding Scheme in star graph, which demonstrates that CXprime successfully detects that star structure frequently appears in all cases. Analogously, in Figure 4.5(b) the Triangle Coding Scheme has a lower coding cost in the triangle graph, which proves that the frequent appearance structure is triangular structure.

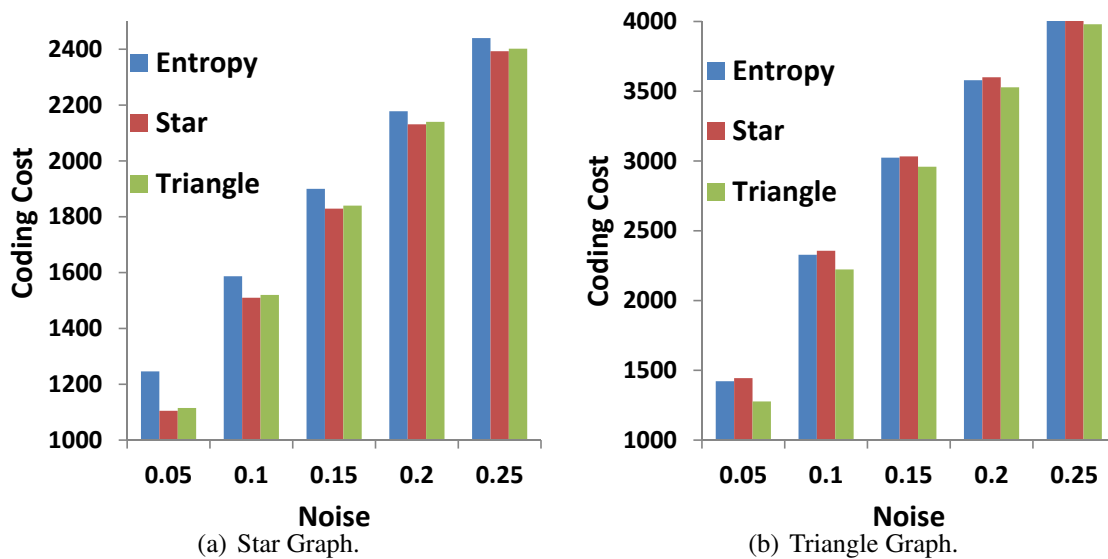


Figure 4.5: Coding Scheme Evaluation on Synthetic Data.

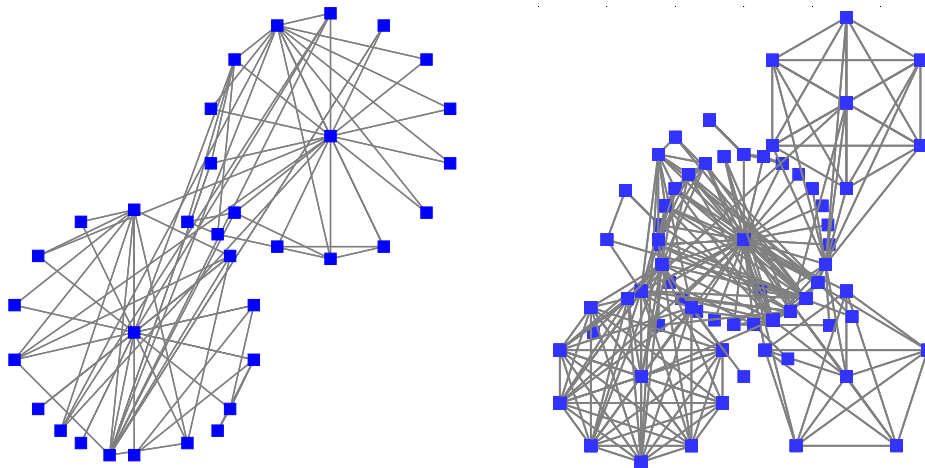
Real Data Sets¹ Our first real-world data set called “Zachary’s karate club” [123] is a social network with 34 nodes which demonstrates the relationship between members of a karate sports club at a US university in the 1970s. The other real world data set “Les Misérables” [60] is a network with 77 nodes indicating characters in Victor Hugo’s famous novel of the same name. The edges are representing the connection between any pair of characters which appear in the same chapter of the book. Both data sets are obtained from the UCI Network Data Repository. Table 4.1 shows the coding costs of “Zachary’s karate club” and “Les Misérables” which calculated by entropy, Star Coding Scheme and Triangle Coding Scheme, respectively. In the case of “Zachary’s karate club” the Star Coding Scheme obtains the minimum coding cost, which implies that this graph is more star-like. In order to further evaluate results of distinguishing graph substructures, we visualize “Zachary’s karate club” data in Figure 4.6(a). Seen from the figure,

¹<http://networkdata.ics.uci.edu/index.php>

there are two striking hubs inside the graph. In terms of “Les Misérables”, comparing with other coding schemes, the Triangle Coding Scheme yields the smallest value, which indicates that there are more triangle structures than star structures in the graph. Moreover, Figure 4.6(b) shows that there are a large amount of obvious triangular structures in the “Les Misérables” network.

Table 4.1: Distinguishing Graph Structure on Real Data Sets

	Entropy	Star	Triangle
Zachary’s karate club	330.9	325.6	331.3
Les Miserables	1251.4	986.6	895.1



(a) Structure of Zachary’s karate club (Star-like). (b) Structure of Les Miserables (Triangle-like).

Figure 4.6: Coding Scheme Evaluation on Real-World Data.

4.4.2 Graph Partitioning

In this section, we compare CXprime with classical graph partitioning algorithms, such as Metis [56] and Markov Clustering (abbreviated as MCL) [112]. Besides, we compare CXprime with Cross-association (abbreviated as CA) [17], which is also a compression-based graph clustering algorithm. Metis and MCL require input parameters. For MCL, we used the default parametrization. CXprime and CA automatically find clusters without any parameter due to their information-theoretic approaches.

Synthetic Data Sets. We generate two synthetic data sets with different structural clusters, and evaluate the graph partitioning performance of CXprime and the comparison methods. We generate each type of data set 10 times using a different random number generator and output the average results.

The first synthetic data set *Syn1* is composed of two star clusters and one clique cluster (100 nodes each) with sketch map shown in Figure 4.7(a). The star cluster is generated with one hub connecting all the other nodes, besides 20 edges are added to it as noise. The clique cluster is created by making a full connected graph first and then 20 edges are removed from it. The edges between each pair of two clusters are randomly selected to connect them, which is why we altered their number from 10 to 100 to evaluate their affects on graph partitioning. Since we know the class label of each node as ground truth, the Normalized Mutual Information (NMI) [114] is used to evaluate the clustering results, and NMI scales between 0 and 1, where 1 means a perfect clustering and 0 means no agreement at all among class and cluster labels. Figure 4.7(b) shows curves of NMI values when implementing CXprime and comparison algorithms on *Syn1* with different number of between edges. Benefiting from finding the structure of the star cluster, CXprime clearly performs better than the other methods with a NMI above 0.9 even when there are 100 edges between each pair of two clusters. Figure 4.8(a) depicts the coding costs of detected clusters in Star Coding Scheme and Triangle Coding Scheme. As expected, Star Coding Scheme gives less bits for two star-like clusters, while Triangle Coding Scheme gives less bits for the clique-like cluster. Metis performs good when there are less edges (below 30) between two clusters, however its performance severely degrades when there are more edges in between (below 0.3 when there are 100 between edges). MCL cannot find correct clusters with a NMI about 0.5 for all the cases. CA fails to detect correct clusters with a NMI no bigger than 0.3, because there is no dense region in a star cluster. Interestingly, CA gets better results when there are more edges in between, which shows that CA can not find sparse clusters.

The second synthetic data set *Syn2* is composed of three star clusters and each star contains 100 nodes, the sketch map is shown in Figure 4.9(a). Similarly, 20 edges are added to each star cluster as noise in order to keep the star structure. Each pair of star clusters is connected with randomly selected edges with their number ranging from 10 to 100. The clustering results of

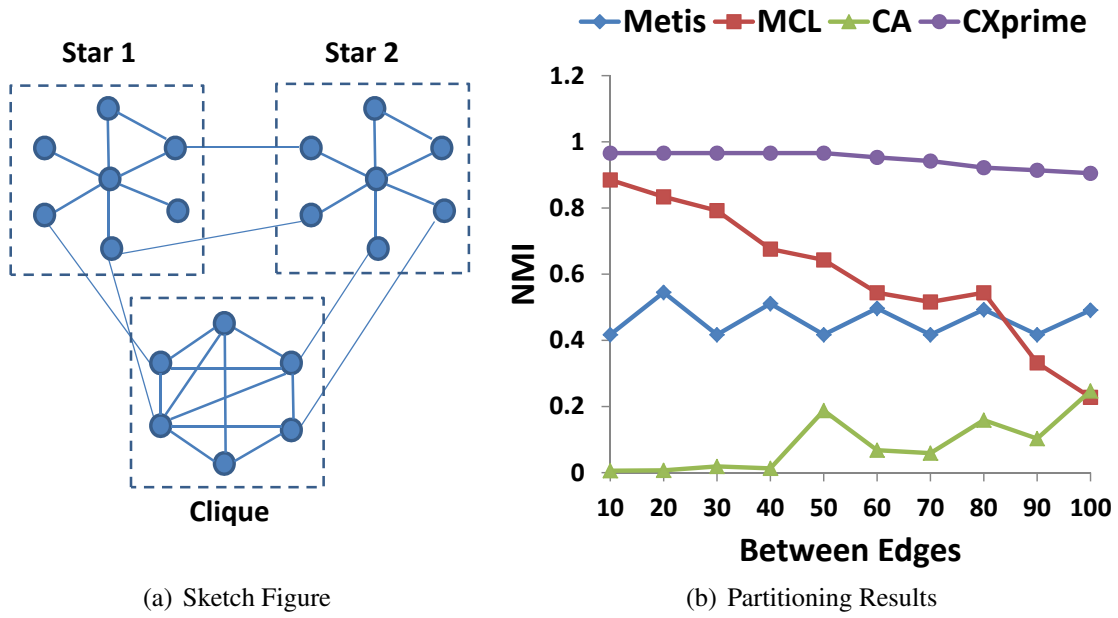


Figure 4.7: Syn1 with Two Stars and One Clique.

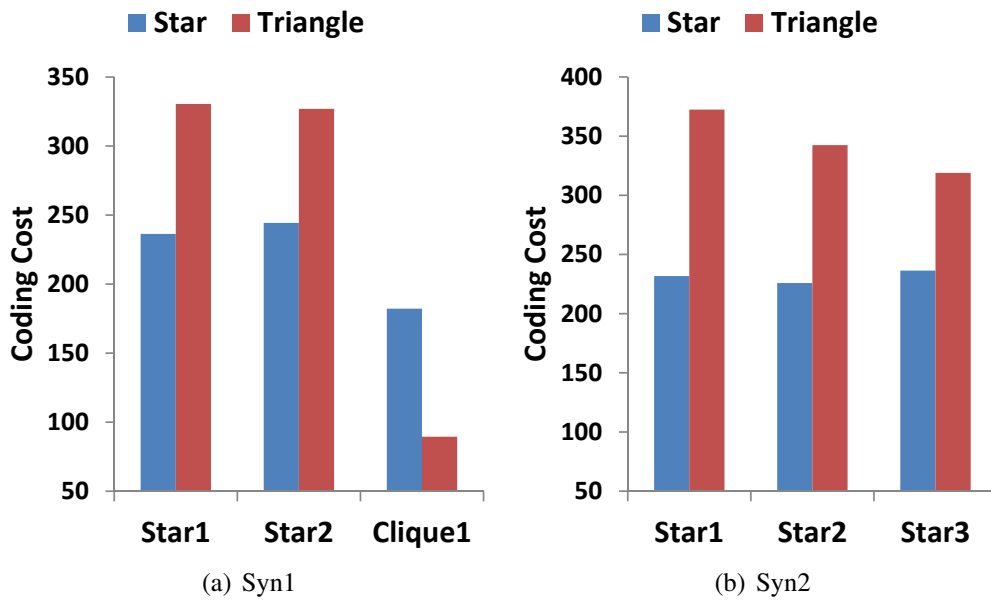


Figure 4.8: Coding Cost of Clusters.

Syn2 which are evaluated by NMI are depicted as curve graph in Figure 4.9(b). Seen from the figure, CXprime clearly performs better than the other methods with a NMI above 0.9 even when there are 100 edges between each pair of two clusters. Figure 4.8(b) depicts the coding costs of detected clusters in Star Coding Scheme and Triangle Coding Scheme, in which the Star Coding Scheme compress all three star-like clusters with less bits. Metis and MCL perform good when there are less edges between clusters, however their performances severely degrades when there are more edges in between. CA cannot detect any clusters in this data set, because CA can not find star-like structures.

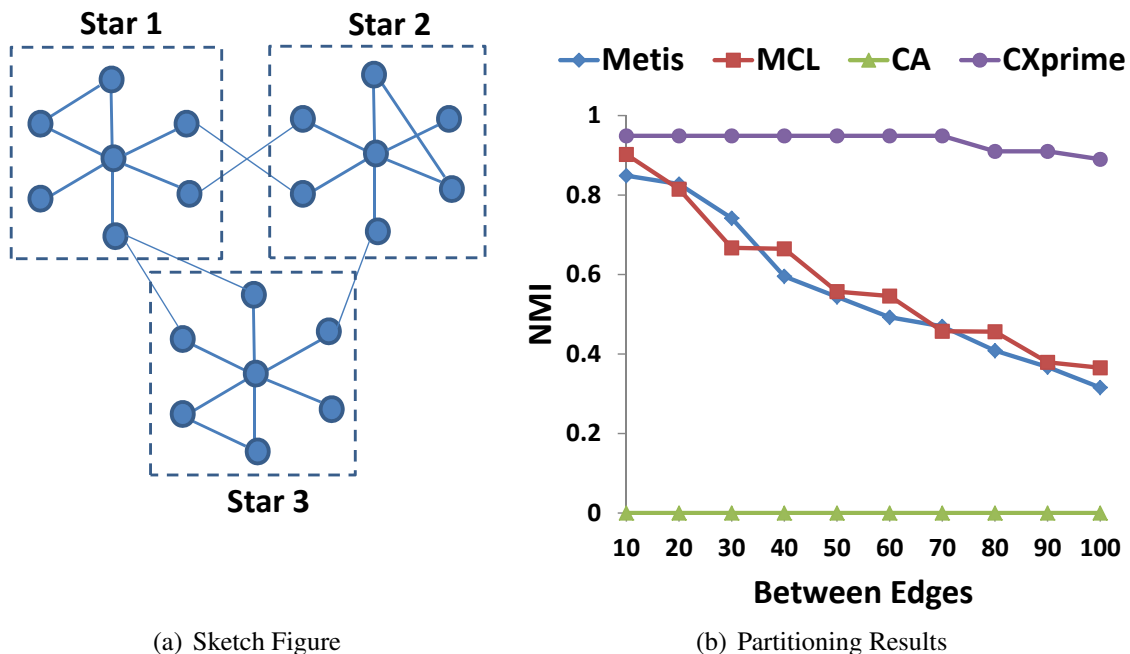


Figure 4.9: Syn2 with Three Stars.

In summary, the results of CXprime are clearly superior than those of the comparison partners on all synthetic data. This demonstrates that CXprime is suitable to detect clusters according to their structure type.

Real Data Sets. Two real data sets are used to evaluate the performance of CXprime on graph partitioning, “Zachary’s karate club” graph and a subgraph of the DBLP co-authorship graph.

“Zachary’s karate club” graph is small and therefore we can assess its structure by directly

drawing it by splitting hub nodes as shown in Figure 4.6(a), which helps us to interpret the partitioning results. CXprime and Cross-association automatically detect two clusters, MCL finds two clusters under the default parameter setting as well. Therefore, we set the cluster number to 2 for Metis. Since class labels are unavailable, we visualize the graph for evaluation. The graph partitioning results are depicted in Figure 4.10, in which different symbols stand for different cluster labels. Seen from Figure 4.10(a), two clusters are detected by CXprime, the one labeled with red triangle is sparser and the one labeled with blue square is denser. Metis and MCL perform well on this data set (Figure 4.10(b) and 4.10(c)), but still have several points grouped differently. However, note that the results of Metis and MCL only consist of the clustering without giving any information on the cluster content. CXprime is the only method providing us not only the clustering but also the interesting information that the content of one clusters is dominated with star structure and the other one contains more triangle due to the dense connection. Cross-association completely fails to detect clusters in such data set as Figure 4.10(d) shows, because there is no very dense region in this graph.

The DBLP² network contains information on which researchers are publishing together and how each research group evolves over time. It has the advantage that we can interpret the results relatively easy based on our personal knowledge and on the knowledge provided open source by DBLP even though the data is unlabeled. We generated our test data set by taking all co-authors of three well-known international professors, namely “Jiawei Han”, “Christos Faloutsos” and “Hans-Peter Kriegel” as nodes and expect the professors to be the nodes with the highest degree (hubs). The co-authors and professors are connected if any two of them cooperated on a paper together. The data set consists of 1014 vertices with 5828 edges between them. In the following we will refer to the three clusters that we expect our comparison methods to find as “han-group”, “faloutsos-group” and “kriegel-group”. Therefore, we set the number of cluster to 3 in Metis, MCL was working with its default parameter and CXprime and CA are both parameter-free algorithms.

The extracted DBLP subgraph does not connect densely, thus CA is not able to find any meaningful clusters. Metis finds three cluster, but one of the clusters contains two hubs “Chris-

²<http://dblp.uni-trier.de/>

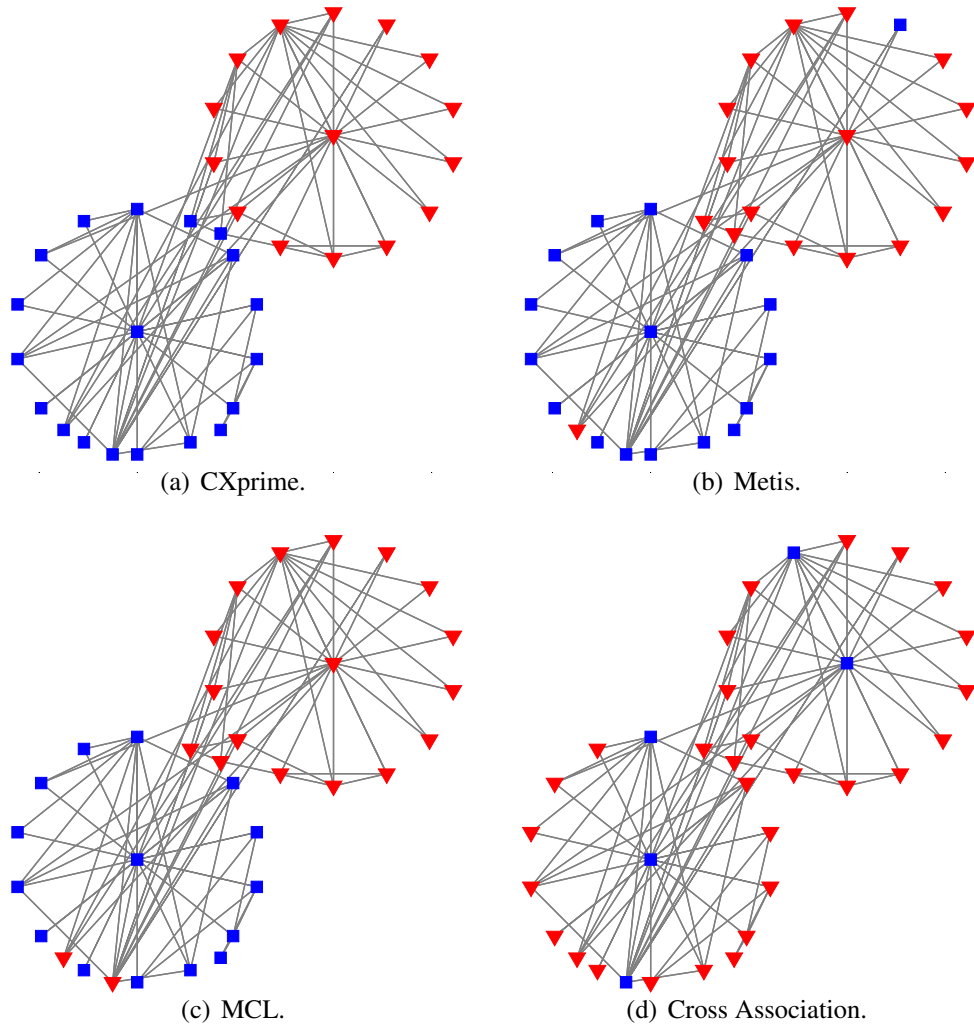


Figure 4.10: Graph Partitioning of Zachary's karate club Data.

tos Faloutsos” and “Hans-Peter Kriegel”, which considerably deviates from the ground truth. The result of MCL and CXprime is similar, all three professors are as expected in different clusters, and in the overall look the quality of both results are good. In detail, they differ in the han-group in 26 people, in the faloutsos-group in 27 people and in the kriegel-group only 8 people. Some examples are shown in Table 4.2. Among these different peoples both methods are not totally correct and most of the differences exist between han-group and faloutsos-group. For example, Hui Zhang and Padhraic Smyth who were falsely put into han-group by MCL. Specifically, Checking from DBLP website, Hui zhang has published two papers with Prof. Faloutsos but has no collaboration with Prof. Han. And Padhraic Smyth also has only collaborated with Prof. Faloutsos. Moreover, Wei-Ying Ma has published one paper with Prof. Kriegel, but is falsely put into han-group by MCL. However, both Senqiang Zhou and John Paul Sondag have published one paper with Prof. Han respectively. But they are falsely grouped into faloutsos-group by CXprime. Therefore, we consider the quality of the results of MCL and CXprime for this dataset as approximately equal.

Table 4.2: Example of Differing People in MCL and CXprime

	MCL	CXprime
Hui Zhang	han-group	faloutsos-group
Padhraic Smyth	han-group	faloutsos-group
Wei-Ying Ma	han-group	kriegel-group
Senqiang Zhou	han-group	faloutsos-group
John Paul Sondag	han-group	faloutsos-group

However, CXprime is the only method detecting the structure of these three clusters. Here, we expect PhDs working at university with the professor to publish in more of a clique structure and external as well as cooperation partners as more of a star one. Also, this evolves over time. CXprime provides the content information of these three clusters that han-group is the biggest group and is more triangle-like. The kriegel-group as the smallest cluster is denser and therefore is formed as triangle-like. And the faloutsos-group is referred to as a more star-like motif, which may caused by containing external students.

Compression Rates. The basic idea of MDL is that the more you compress the data the more you

learn from it. Therefore, it is non-trivial to compare CXprime with compression-based methods in terms of compression rates. We compare the compression rates between these methods: our Star Coding Scheme and Triangle coding scheme, our CXprime partition algorithm, and two existing compression-based graph mining algorithms SLASHBURN[54] and Cross-association (CA), and entropy is given as a base line. The results for both synthetic (*Syn1* and *Syn2* with 10 edges between each pair of clusters) and real datasets are depicted in Table 4.3, which are shown in bits. As only half of the adjacency matrix is considered in this chapter, the compression rates of SLASHBURN and CA are also calculated from the half of the matrix. And the number in bracket are the sizes of blocks for SLASHBURN, we try different settings and output the best compression rate. It is clear that CXprime outperforms the other methods by achieving higher compression rates in both synthetic and real data sets.

Table 4.3: Compression Rates (Bits)

	Entropy	Star	Triangle	Cxprime	SLASHBURN	CA
Syn1	23215	12906	4345	1397.28	4056.6(20)	2230.5
Syn2	3114	1926.8	1930.5	1390.9	2399.1(20)	1695.5
Karate	330.9	325.6	331.3	297.1	306.6(11)	317
DBLP	46027	44743	39249	35297	38538(50)	38485

4.4.3 Link Prediction

Considering the structures which can be distinguished by CXprime, an unsupervised link prediction score is proposed. In this chapter, we compare our proposed score with other unsupervised link prediction scores, Common Neighbors (CN), Preference Attachment (PA) and Katz ($\beta = 0.005$). All scores are experimented on both synthetic and real data sets. In order to evaluate the efficiency of our scores, we randomly sample 30% of edges as predicting edges S and deleted them from the original graph. In the resulting graph, we calculate the link-prediction scores of every pair of unconnected nodes and sort them descending. The first $|S|$ edges of each score are selected separately as a predicted result which is expressed as P . After comparing the predicting edges S with predicted edges P , we use the precision $|S \cap P|/|S|$ to evaluate the

results. All results are the average values of 10 times running the algorithms.

Synthetic Data Sets. We implement the link prediction scores on both triangle graph and star graph which are generated in the same way like the synthetic data sets in section 4.4.1. The precisions of four unsupervised link prediction scores of the triangle graph and star graph with percentage of noise edges ranging from 0.05 to 0.25 are shown in Figure 4.11(a) and Figure 4.11(b) separately. Clearly, Figure 4.11(a) shows that CXprime possesses more or less higher precision in a triangle graph than the other three scores in each cases. Moreover, seen from Figure 4.11(b), CXprime occupies the highest position of the four scores in first four cases of star graph. Especially when the noise density is small, the advantage of CXprime is more remarkable.

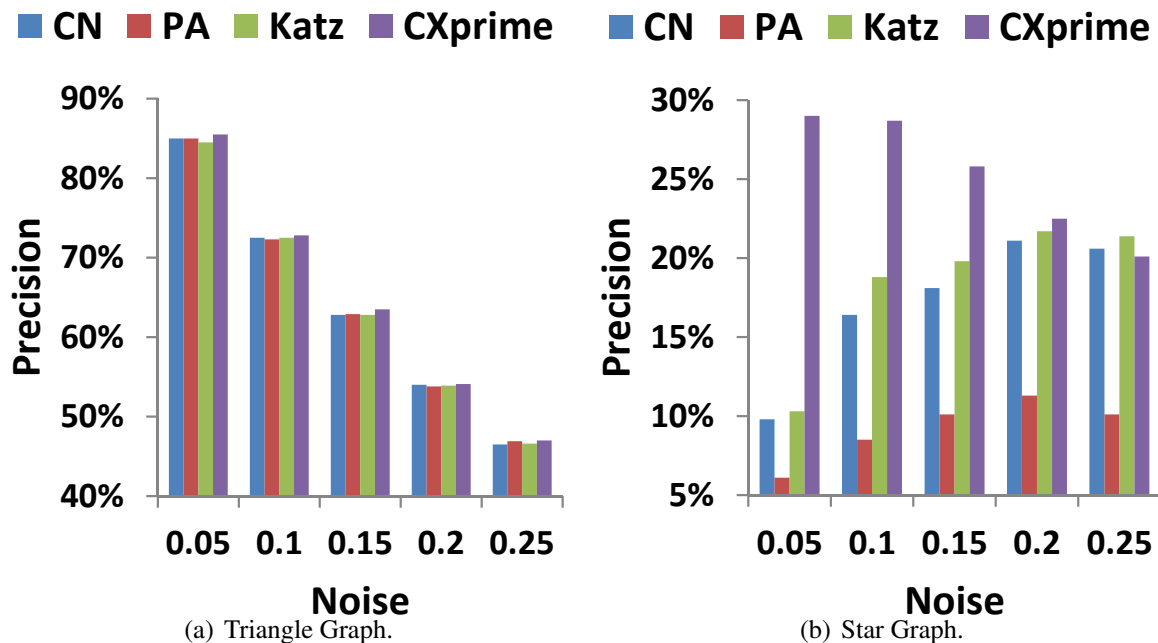


Figure 4.11: Precision of Synthetic Data Sets.

Real Data Sets.³ “Zachary’s karate club” and “Copperfield Word Adjacencies”[83] are adopted in this part. “Copperfield Word Adjacencies” is the network with 112 nodes and 425 edges which represent common adjective and noun adjacencies for the novel “David Copperfield” by Charles Dickens. The link prediction precision of each score is displayed in Table 4.4 which shows that our proposed score of CXprime outperforms the other three scores of the given comparison

³<http://networkdata.ics.uci.edu/index.php>

methods on these two real data sets.

Table 4.4: Precision of Real Data Sets (%)

	CN	Katz	PA	CXprime
Zachary’s karate club	16.9	15.6	16.5	26.1
Copperfield Word Adjacencies	11.1	16.5	11.5	16.7

4.5 Related Work and Discussion

We briefly survey related work on four relevant topics addressed by CXPrime: graph structure and pattern mining, compression-based graph mining, graph partitioning and link prediction.

4.5.1 Graph Structure and Pattern Mining

Research on the structure of complex networks has already gained significant attention. Most real world graphs follow power-law degree distributions, which can distinguish between an actual real-world graph and any artificial one [16]. In a power-law graph, most vertices have a very low degree, while few ones have extremely high degrees (we call this pattern a star). The existence of these hubs makes the community detection in these real-world graphs very difficult, because most existing graph clustering techniques focus on densely connected communities (triangle-types). On the other hand, there are many frequent subgraph mining or motif detection algorithms, [120, 98] to mention a few, which aim to find recurrent and statistically significant sub-graphs. Our technique CXprime combines these two approaches and exploits structure primitives to discover the graph structure itself.

4.5.2 Compression-based Graph Mining

Due to the increasing scale of graph data, there are many algorithms proposed for efficiently compressing an graph, e.g.[10, 19] to mention a few. However, the compression aspect is not the major focus in this chapter, but knowledge discovery from an information-theoretic background.

SUBDUE [50] and Cross-association [17] are two famous algorithms also relying on the Minimum Description Length principle to identify patterns in a graph. SUBDUE uses a heuristic search guided by MDL to find specific patterns minimizing the description length of the entire graph. Other than CXprime, SUBDUE is not able to find the global general structure of a graph. Cross-association is another co-clustering method, which can be applied for unipartite graphs as well, processing a binary matrix and seeking clusters of rows and columns. Then the matrix is divided into homogeneous rectangles which are representing the underlying structure of the data. Cross-association can only find dense triangle communities and is therefore not suited for many sparse real graphs, while CXprime is explicitly designed for such types of graphs. Another, more recent work called SLASHBURN [54] was proposed for graph compression exploiting the hubs and the neighbors of hubs. SLASHBURN uses the power-law characteristic for compression, and can only exploit dense communities.

4.5.3 Graph Partitioning

There are plenty of works on graph clustering or graph partitioning, including spectral clustering [99], Min-Max-Cut algorithms[26], Metis [56], Markov Clustering [112] for unipartite graphs, co-clustering [25], Cross-association [17] and SCMiner [32] for bipartite ones. All these methods aim to find regions with more intra edges than inter edges, in which densely connected subgraph communities can be found. However, none of them considers interesting sparse patterns like stars, which are prevalent in real-world graphs. Therefore, these approaches fail in detecting star-like communities, while the proposed CXprime can distinguish between star-like and triangle-like sub-structures.

4.5.4 Link prediction

Inferring whether two disconnected nodes will be linked in the future, based on available graph information is the task of link prediction in graph mining. Liben-Nowell and Kleinberg [66] summarize some unsupervised link prediction approaches for social networks. For example, common neighbour, preferential attachment [75], Katz [58] and others. Specifically, preferential

attachment is based on the idea that two nodes with higher degrees have a higher probability to be connected. Katz defines a score that sums up the number of all paths between two nodes where short paths are weighted stronger. Obviously, common neighbour and Katz are more effective in triangle graphs, while preferential attachment works better in star-like graphs. However, none of the previous methods consider the underlying graph structure information which is important for link prediction. To the best of our knowledge, CXprime is the first method that is using structure information to improve the quality of link prediction.

4.6 Chapter Conclusion

In this chapter, we introduced CXprime, an multi-functional algorithm for mining graphs based on structure primitives. The two key ideas of CXPrime are to model the transitivity and the hubness of a graph using three-node primitives and to exploit the relative frequency of these structure primitives for data compression and knowledge discovery. We demonstrate that the combination of these two ideas is very useful for (1) automatically detecting the structure type of a graph as star-like/scale-free or clique-like/small-world, (2) clustering the graph into homogeneously structured sub-graphs and (3) accurate link prediction. Our experiments demonstrate that the knowledge about the structure type is very interesting for interpreting graphs and that our novel structure-based cluster notion is a valuable complement to traditional graph clustering methods searching for dense sub-networks only. Regarding link-prediction, we outperform existing methods especially on star-like graphs which demonstrates that the knowledge about the structure type is indeed very useful for graph mining.

Chapter 5

Summarization-based Mining Bipartite Graphs

In this chapter, our research target is transferred from the simple graph to bipartite graph. Bipartite graph is suitable to model the relation between two classes of objects, such as customs and items, people and companies. In terms of clustering, many algorithms named as co-clustering or bi-clustering are proposed to detect clusters from such two relational data. However, most of them focus on finding out clusters of each class of objects simultaneously but ignore analyzing the relationship between the clusters. In this chapter, we model such data sets as bipartite graph. From the perspective of structure, we not only obtain clusters of each class of objects but also achieve other graph mining tasks at the same time, such as graph summarization, discovery of the hidden structure and link prediction. We propose an algorithm named as SCMiner (**Summarization-Compression Miner**). The core idea of the algorithm is compressing a large bipartite graph to a highly compact representation. And the progress is achieved by merging nodes with same link pattern after adding or removing links based on some strategies. Guided by the Minimum Description Length principle, our technique is able to achieve good clustering result, to discover the relation between the clusters and to predict the missing or suspicious links. And all the tasks can be achieved automatically and do not rely on any input parameters which are difficult to estimate. Comparing with the state-of-the-art techniques, experiments on synthetic

and real data demonstrate the advantages of SCMiner.

The remainder of this chapter is organized as follows: Section 5.1 introduces the motivations and contributions. Section 5.2 gives the model formulation and coding scheme. Section 5.3 presents the algorithm in more detail. Section 5.4 contains an extensive experimental evaluation. Section 5.5 discusses related work and Section 5.6 gives the conclusion of this chapter.

Parts of the material presented in this chapter have been published in [32], where Jing Feng was responsible for the development of the main concept and wrote the largest parts of the paper, Xiao He also contributed to concept development and implementation; Claudia Plant supervised the project and made contributions to the building of coding scheme; Bettina Konte performed part of experiments; Christian Böhm revised the whole paper; The co-authors also contributed to the conceptual development and paper writing.

“Jing Feng, Xiao He, Bettina Konte, Christian Böhm and Claudia Plant. Summarization-based Mining Bipartite Graphs. The 18th ACM SIGKDD Conference on Knowledge Discovery and Data Mining 2012: 1249-1257.”

5.1 Introduction

5.1.1 Motivation

Relational or graph-structured data are prevalent in nature, science and economics. Many aspects of real life can be well represented as a bipartite graph which has two types of vertices and edges representing the connections between them. Consider for example newsgroups where one type of vertices represents persons and the other type of vertices represent groups. An edge between a person and a group means that he or she is member of this group. Or consider the interaction between drugs and proteins: An edge between a particular substance and a protein means that the corresponding protein is responding to the drug. Such bipartite graphs are stored as large adjacency matrices often having millions of vertices and edges. Effective and efficient data mining methods are essential for answering high-level domain specific questions like: which

users are potentially interested to join a newsgroup? Or, which combination of substances is most effective against a certain disease?

Therefore, in recent years the research topic of mining bipartite graphs has attracted much attention, with a large volume of research papers, e.g. [17, 23, 36, 66, 67, 81] to mention a few. To extract knowledge from a large bipartite graph, existing techniques apply one of the following two basic strategies: (1) Clustering, as shown in Figure 5.1 (a). These approaches reduce the complexity by partitioning the large input graph into smaller groups of similar vertices which can be inspected and interpreted by the user. In particular, approaches to co-clustering like [17, 23] cluster both types of vertices, i.e. the rows and the columns of the adjacency matrix simultaneously guided by the idea that the clustering of rows and columns can profit from another. The output of co-clustering is a set of row-clusters and a set of column clusters. (2) Summarization, as shown in Figure 5.1 (b). These approaches reduce the complexity not by grouping but by a global abstraction. The output of summarization techniques like [81] is a bipartite graph which is much smaller than the original input graph. Ideally, it distills the major characteristics from the input data and is small enough to be accessible for the user. (3) Link prediction, as shown in Figure 5.1 (c). These approaches like [67] study the question whether there should be an edge between two currently disconnected vertices. Link prediction is closely related to summarization and clustering since a technique for link prediction must capture at least the local structure of the graph to provide reasonable predictions. Also clustering and summarization are closely related since during the process of abstraction, both approaches must identify the relevant major characteristics of the graph.

5.1.2 Contributions

In this chapter, we therefore propose a bipartite graph mining technique named as SCMiner. As shown in Figure 5.1, SCMiner is the algorithm that integrating the tasks of graph clustering, graph summarization and link prediction on bipartite graphs. The name SCMiner stands for **S**ummarization **C**ompression **M**iner and the principle of data compression also known as the Minimum Description Length Principle (MDL) [92] is the basis of our technique. The basic

idea is to transform the original graph into a very compact summary graph. During the process of transformation which is controlled by the MDL principle, our technique discovers the major clusters of both vertex types as well as the major connection patterns between those clusters. The result of SCMiner comprises a compressed graph, the row- and column-clusters and their link patterns. The major contributions of our approach can be summarized as follows:

- **Clustering plus hidden structure mining.** Like state-of-the-art co-clustering methods, SCMiner accurately identifies the row- and column clusters of bipartite graphs and even outperforms them on some data sets. As a key feature of our approach, the result does not only consist of two sets of clusters. SCMiner also reveals the relationships between the clusters which are essential for interpretation.
- **Accurate link prediction.** SCMiner accurately predicts missing or future links and removes noise edges.
- **Unsupervised graph mining all in one.** SCMiner integrates summarization, clustering and link prediction and thus comprehensively supports unsupervised graph mining.
- **Results validated by data compression.** Data compression is an intuitive optimization goal with many benefits: By natural balancing goodness of fit and model complexity, overfitting is avoided. The results are thus simple and interpretable. Moreover, supported by the MDL principle SCMiner does not rely on any difficult to estimate input parameters.

5.2 Compressing a Bipartite Graph

In this section, we first present a simple example to introduce the terminology of graph summarization. Then an MDL based coding schema is derived to find the best summarization of a bipartite graph. At last, we propose a strategy to discover hidden relations between vertices of different type.

Notation and Example. Figure 5.2 depicts a simple example for graph summarization of a bipartite graph. $G = (V_1, V_2, E)$ is an unweighted bipartite graph where $V_1 = \{V_{11}, \dots, V_{16}\}$

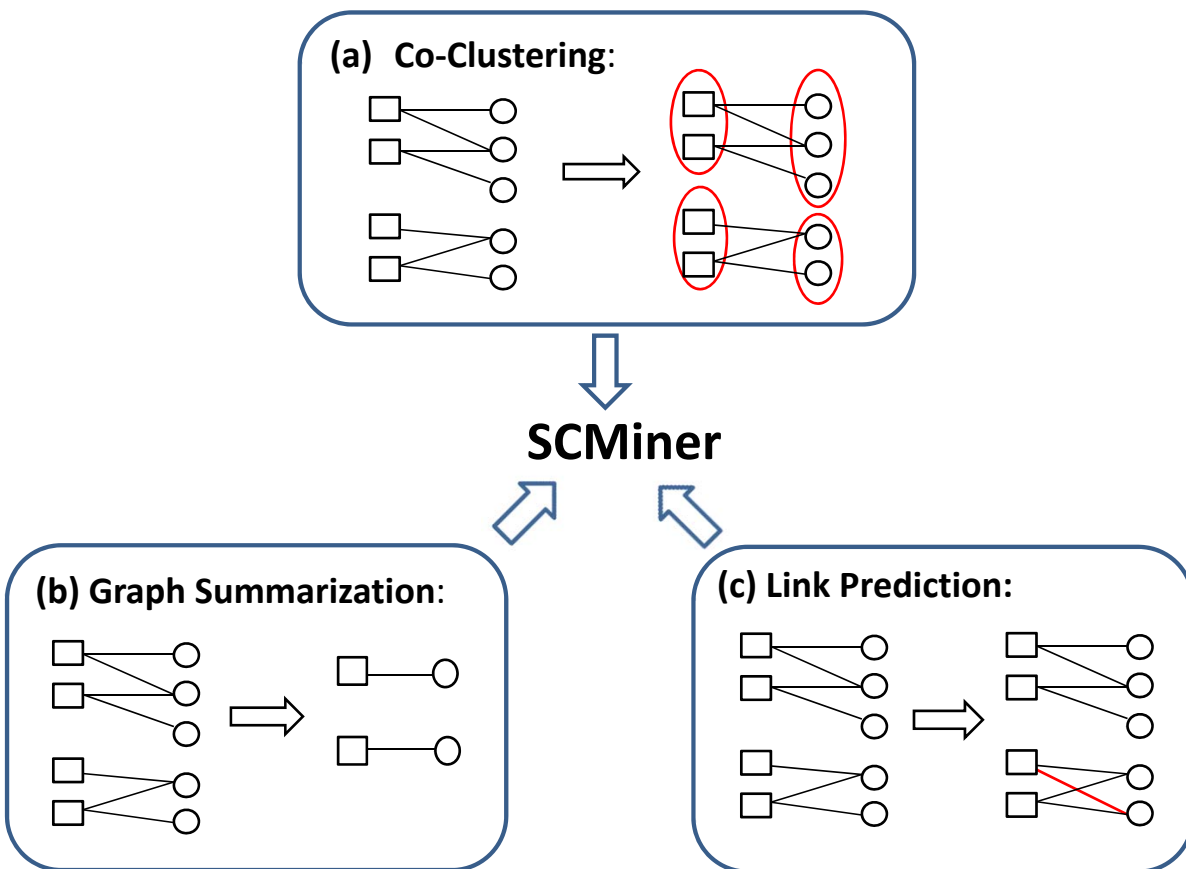


Figure 5.1: Tasks of SCMiner.

and $V_2 = \{V_{21}, \dots, V_{26}\}$ denote two types of vertices and E denotes the edges between them. The summarization of graph G consists of two bipartite graphs, a summary graph G_S and an additional graph G_A . The summary graph $G_S = (S_1, S_2, E')$ is an aggregated graph and is composed of four super nodes $S_{11} = \{V_{11}, V_{12}, V_{13}\}$, $S_{12} = \{V_{14}, V_{15}, V_{16}\}$, $S_{21} = \{V_{21}, V_{22}, V_{23}\}$, $S_{22} = \{V_{24}, V_{25}, V_{26}\}$, and super edges E' . The super nodes themselves are composed of vertices exhibiting the exact same link pattern. They indicate the local cluster structure of each type of vertices, whereas the super edges represent the global relations between local clusters. The additional graph $G_A = (V_1, V_2, E'')$ contains normal nodes and correction edges which are required to reconstruct the bipartite graph G . The $+$ or $-$ symbols above the edges indicate whether it is required to add or remove them from G_S to recreate G . The correction edges describe the revealed hidden relations between different types of vertices.

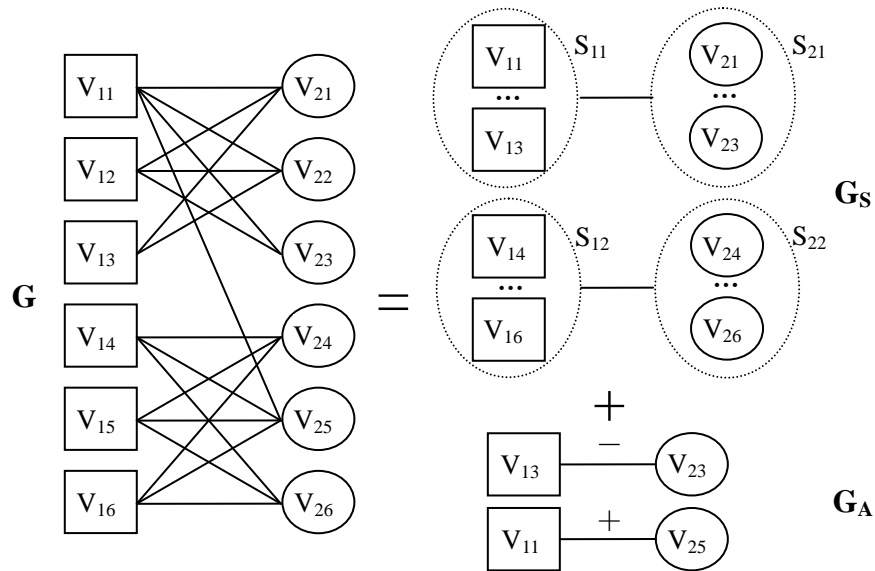


Figure 5.2: Summarization and Compression.

5.2.1 Coding Scheme

A bipartite graph can be represented by thousands of summarizations, however, the challenge is to find out the one that best represents the data and reflects the hidden structure behind the surface data. The Minimum Description Length (MDL) principle is an intuitive choice for model

selection [92]. It follows the assumption that the more we are able to compress the data, the more we have learned about its underlying patterns. Formally, the goodness of the model can be stated as shown in Eq.(5.1), where $L(M)$ denotes the cost for coding the model parameters and $L(D|M)$ represents the cost of describing the data D under the model M . As the model has to be coded with the data and too complex models result in high compression cost, MDL naturally avoids overfitting.

$$L(M, D) = L(D|M) + L(M). \quad (5.1)$$

Inspired by the MDL principle, we propose a coding scheme to choose the best graph summarization. The most direct and simple way to represent a bipartite graph $G = (V_1, V_2, E)$ is to compress its adjacency matrix $\mathcal{A} \in |V_1| \times |V_2|$, with $\mathcal{A}_{i,j} = 1$ if $(V_{1i}, V_{2j}) \in E$. The coding cost of the adjacency matrix \mathcal{A} is lower bounded by its entropy.

As mentioned above, the summarization of a bipartite graph G is composed of two bipartite graphs, the summary graph G_S and the additional graph G_A . Instead of transferring G from a sender to a receiver, we can transfer G_S , G_A and the group information of super nodes in G_S . In addition, there is no need to send the symbol information of edges in G_A , since this information can be obtained by comparing G_A and G_S . If the symbol of an edge in G_A is $+$, this edge is not present in G_S , and vice versa, if the symbol is $-$, G_S contains the edge. The following equation defines the coding cost of a summarization of the graph.

Definition 5.1 (Coding Cost of a Graph.)

$$CC(G) = CC(G_S) + CC(G_A) + CC(group), \quad (5.2)$$

$$CC(group) = \sum_{i=1}^{N_T} \sum_{j=1}^{N_i} |S_{ij}| \log_2 \frac{|V_i|}{|S_{ij}|}, \quad (5.3)$$

where $CC(G_S)$ and $CC(G_A)$ denote the coding cost of summary graph G_S and the additional graph G_A . The third term is the cost of coding the group information, where N_T is the number of node types, N_i is the number of super nodes in type i , $|S_{ij}|$ is the number of members in super

node S_{ij} , $|V_i|$ is the number of original nodes in type i .

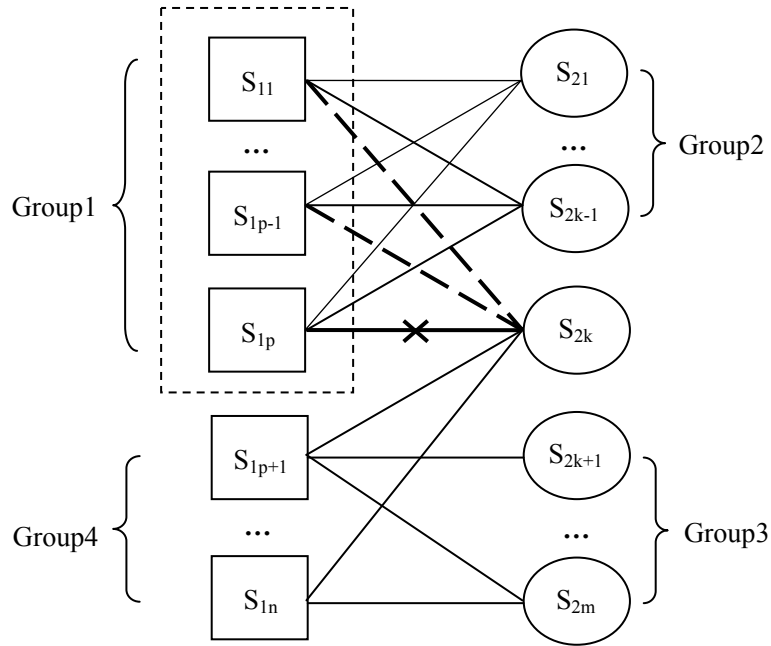
$L(D|M) = CC(G_S)$ is the description of the data under the model M with coding cost $L(M) = CC(G_A) + CC(group)$. The optimization goal of our algorithm SCMiner is to find the best model or summarization that minimizes Eq.(5.2).

5.2.2 Hidden Relations Between Vertices

The data we record in real life applications does often only approximately represent the ground truth due to inconsistencies and measurement errors. For example, the same user often joins newsgroups under different nicknames and email addresses. Or a protein might show a strong response to a substance simply due to a measurement error. Thus real world graphs are often spoiled by erroneous connections on the one hand and are also missing important links on the other hand.

Take a close look at the graph G in Figure 5.2, obviously we have the feeling that edge (V_{13}, V_{23}) is probably missing, and edge (V_{11}, V_{25}) maybe presents an artifact of noise. Therefore, if we add edge (V_{13}, V_{23}) to G and remove edge (V_{11}, V_{25}) from G , we can form four super nodes, whose members exhibit the exact same connection patterns. These connections probably reflect the true relationships. Based on these observations, we propose the following strategy to discover the hidden relations between vertices.

Figure 5.3 describes our strategy for merging nodes, suppose all the nodes are super nodes. The nodes in Group 1 share a similar link pattern and could therefore be merged into a super node. Nodes in Group 2 and Group 3 are both hop two neighbors of node S_{2k} , specifically nodes in Group 2 are common neighbors of nodes in Group 1 and nodes in Group 3 are not. To form a new super node of nodes in Group 1 their link pattern should be exactly the same. However, the link pattern of node S_{1p} in Group 1 is similar to those of the other nodes, but not the same. Concretely, S_{1p} has an extra link with S_{2k} that the other nodes lack. As we can see from Figure 5.3, two methods can be adopted to make nodes in Group 1 exhibiting the same link pattern: we could either remove edge (S_{1p}, S_{2k}) or add edges $(S_{11}, S_{2k}), (S_{12}, S_{2k}), \dots, (S_{1p-1}, S_{2k})$. The question is, how to distinguish whether edges should be removed or added?



Group1: Nodes to be merged

Group2: Hop 2 neighbors of S_{2k} and common neighbor of Group1

Group3: Hop 2 neighbors of S_{2k} but not common neighbor of Group1

Group4: Neighbors of S_{2k}

~~—~~ : The edge to be deleted

- - - : The edge to be added

Figure 5.3: The Strategy Used for Merging Nodes

Algorithm 5.1 *ModifyEdge*

Input: Group nodes $group$, G_S , G_A

Output: G_S , G_A , $hop2Sim$

//Modify edges of group to make their link same:

- 1: $alln = Neighbor(group)$;
 - 2: $cn = CommonNeighbor(group)$;
 - 3: **for** Each node $S \in alln$ and $S \notin cn$ **do**
 - 4: Using Eq.(5.4) and Eq.(5.5) to add or remove edge;
 - 5: Add or remove edges in G_S ;
 - 6: Add additional edges to G_A ;
 - 7: **end for**
 - 8: Update $hop2Sim$ for each $S \in alln$ and $S \notin cn$;
 - 9: **return** G_S , G_A , $hop2Sim$;
-

We can decide if we add or remove edges by calculating some cost function. We define the cost as the number of edges we add or delete, which is highly related to the MDL costs of Eq.(5.2), in order to make the link patterns of merging nodes equal. Specifically, as shown in Figure 5.3, there are p nodes in Group 1 $S_{11}, S_{12}, \dots, S_{1p}$, among which S_{1p} exhibits a similar but not exact same link pattern as the other nodes. Therefore, the cost of removing edge (S_{1p}, S_{2k}) is 1 and the cost of adding edges $(S_{11}, S_{2k}), (S_{12}, S_{2k}), \dots, (S_{1p-1}, S_{2k})$ is $p - 1$. Obviously, the former method should be selected because of the lower cost. The pseudocode of modifying the edges of a merging group is shown in Algorithm 5.1. At first we combine neighbors and common neighbors of nodes in merging groups, then we need to modify the edges of those nodes which are neighbors but not common neighbors of the merging group to make the link pattern of all group members exactly the same. Suppose the merging group contains $S_{11}, S_{12}, \dots, S_{1p}$ and S_{2k} is one of their neighbors but not a common neighbor. The cost of removing and adding edges can be calculated by Eq.(5.4) and Eq.(5.5). The cost of a super edge is calculated as $|S_{1i}| \cdot |S_{2k}|$, where $|\cdot|$ is the number of normal nodes contained in a super node. However, in some cases the costs for removing might be the same as for adding edges. If so, we use the similarity proposed in the next section to decide which action to take. Naturally, if S_{2k} is more similar to the nodes of Group 2 compared to those of Group 3, we add edges for merging, while otherwise we remove edges.

$$Cost_{remove} = \sum_{i=1}^p \begin{cases} |S_{1i}| \cdot |S_{2k}| & \text{if } S_{1i} \text{ links to } S_{2k} \\ 0 & \text{otherwise.} \end{cases} \quad (5.4)$$

$$Cost_{add} = \sum_{i=1}^p \begin{cases} 0 & \text{if } S_{1i} \text{ links to } S_{2k} \\ |S_{1i}| \cdot |S_{2k}| & \text{otherwise.} \end{cases} \quad (5.5)$$

5.3 Algorithm SCMiner

In this section, we propose our algorithm SCMiner to find the best summarization of a bipartite graph. It can be proven that finding the global optimal summarization is NP-hard, therefore SCMiner follows a heuristic approach to search the local optima.

5.3.1 Basic Idea

The idea behind the algorithm SCMiner is that nodes with similar link patterns can be merged to groups, if the similarity between each pair of nodes in the group is bigger than some threshold th . By adding and removing edges as proposed in Section 5.2 we accomplish that all group nodes share the exact same link pattern and so can form a super node. SCMiner iteratively merges groups of nodes or super nodes whose similarities are bigger than th . The initial threshold is 1 and th is reduced stepwise by ϵ when no pair of nodes can be merged. In each iteration we calculate the new summarization coding cost. Then the MDL principle is used to choose the best summarization. The algorithm terminates when th reaches 0.

5.3.2 Finding Super Node Candidates

To find suitable candidate groups of nodes where merging might pay-off, we introduce the notion of hop two similarity. Suppose two super nodes S_{1i} and S_{1j} have n common neighbors $\{S_{21} \dots S_{2n}\}$ and m neighbors $\{S_{2(n+1)} \dots S_{2(n+m)}\}$ that are connected to only one of the two super nodes. Intuitively, if the two nodes have more common neighbors than neighbors only linking to one node, they are more similar in link pattern, then different. We can define their similarity as

Definition 5.2 (Hop Two Similarity.)

$$sim(S_{1i}, S_{1j}) = \frac{\sum_{k=1}^n |S_{2k}|}{\sum_{k=1}^{n+m} |S_{2k}|} \quad (5.6)$$

where $|S_{2i}|$ is the number of normal nodes contained in super node S_{2i} . This similarity measure ranges from 0 to 1.

As described above, we only need to calculate the similarity between two nodes if they share at least one common neighbor and therefore are hop two neighbors of each other. In the following we call the similarity between a node and all its hop two neighbors its hop two similarity.

Algorithm 5.2 SCMiner

Input: Bipartite graph $G = (V, E)$, Reduce step ϵ **Output:** Summary Graph G_S , Addition Graph G_A **//Initialization:**

- 1: $G_S = G, G_A = (V, \emptyset)$;
- 2: Compute $mincc$ using Eq.(5.2) with G_S and G_A ;
- 3: $bestG_S = G_S, bestG_A = G_A$;
- 4: Compute $hop2Sim$ for each $S \in G_S$ using Eq.(5.6);

//Searching for best Summarization:

- 5: **while** $th > 0$ **do**
 - 6: **for** each node $S \in G_S$ **do**
 - 7: Get SN with $S' \in SN$ and $hop2Sim(S, S') > th$;
 - 8: **end for**
 - 9: Combine SN and get non-overlapped groups $allgroup$;
 - 10: **for** each $group \in allgroup$ **do**
 - 11: **ModifyEdge**($group, G_S, G_A$);
 - 12: Merge nodes of G_S with same link pattern;
 - 13: Compute cc using Eq.(5.2) with G_S and G_A ;
 - 14: Record $bestG_S, bestG_A$, and $mincc$ if $cc < mincc$;
 - 15: **end for**
 - 16: **if** $allgroup == \emptyset$ **then**
 - 17: $th = th - \epsilon$;
 - 18: **else**
 - 19: $th = 1.0$;
 - 20: **end if**
 - 21: **end while**
 - 22: **return** $bestG_S, bestG_A$;
-

5.3.3 Algorithm

Now we describe our algorithm SCMiner, the pseudocode is provided in algorithm 5.2. The input parameters of SCMiner are the bipartite graph $G = (V, E)$, where $V = (V_1, V_2)$ and E are the edges between V_1 and V_2 , and stepsize ϵ for reducing th . The output of SCMiner is the summarization of G , including the summary graph $G_S = (S, E_S)$ composed of super nodes $S = (S_1, S_2)$ and the additional graph $G_A = (V, E')$ composed of single nodes $V = (V_1, V_2)$, which has the minimum coding cost regarding the proposed coding scheme. In the initialization phase, we first set the summary graph G_S to the input graph G , which means that all single nodes are treated as super nodes containing only one normal node, and set the additional graph G_A empty. Then we initialize the coding cost $mincc$ using Eq.(5.2) with G_S and G_A . Afterwards we compute the similarities between each node and its hop two neighbors of same node type. In the searching phase, when $th > 0$, we do the following steps: First we search for groups of nodes that have at least one hop two neighbor with similarity larger than th and then we merge every group we found. When there is no more group of nodes that can be merged, we decrease the threshold th by ϵ . In the merging phase, we use the proposed method shown in algorithm 5.1 to modify the edges of the merging group that are present in G_S to get nodes with exact same link structure. Subsequently we add the corresponding additional edges to G_A and update the hop two similarity for affected nodes, which are neighbors of merging group nodes but not their common neighbors. During the merging phase, nodes with similar link patterns are merged, which decreases the data cost and increases the model cost, while the total cost is reduced in most cases. After each merging step, we calculate the coding cost using Eq.(5.2) with current G_S and G_A and the best summarization with minimum coding cost is stored in $bestG_S$ and $bestG_A$. Finally, we output $bestG_S$ and $bestG_A$ with coding cost.

5.3.4 Properties

We adjust the parameter ϵ using the MDL principle. Extensive experiments on synthetic and real data showed that a suitable range for ϵ is around 0.01 to 0.1. Therefore we try out every setting with a step size of 0.01 and let MDL select the best result. The runtime complexity for

the computation of the similarity between each vertex v and its hop two neighbors is $O(|V| \cdot d_{av}^3)$, where $|V|$ is the number of vertices and d_{av} is the average degree of each vertex. During each merging step in SCMiner, we only compute the similarities between some affected vertices and their two hop neighbors. Therefore the runtime complexity is roughly $O(d_{av}^4)$. The number of merging steps, affected by the reduction stepsize ϵ , is N in average. So the whole runtime complexity is $O(|V| \cdot d_{av}^4)$.

5.4 Experiments

This section provides empirical evidence to show the effectiveness of SCMiner on synthetic and real data. In particular, we evaluate SCMiner three aspects: First, we compare SCMiner with state-of-the-art co-clustering algorithms in terms of quality of the clusters detected on each type of nodes. Secondly, we evaluate the hidden structure, i.e. the relationships between both types of nodes found by SCMiner. Finally, we compare SCMiner with some state-of-the-art link prediction methods to evaluate the validity of hidden relationships discovered by SCMiner.

5.4.1 Data Sets

The generated synthetic bipartite graphs with different parameters are shown in Table 5.1. Both V_1 and V_2 contain the same number of clusters, and each cluster has 100 nodes. The matrix T shows the ground truth links between clusters of different type, where T_{pq} can take the values 1 or 0, which means the p th cluster of V_1 and the q th cluster of V_2 are fully connected or separated. The matrix S introduces link parameters to matrix T , where S_{pq} denotes the percentage of links generated between the p th cluster of V_1 and the q th cluster of V_2 . In other words, if link parameters are introduced based on 0, noisy links are added to the ground truth graph. If link parameters are introduced based on 1, links are removed from the ground truth graph, where the percentage is 1 minus the link parameters.

Three real data sets are evaluated in our experiments. World cities¹ data consists of the dis-

¹<http://www.lboro.ac.uk/gawc/datasets/da6.html>

Table 5.1: Synthetic Bipartite Graphs

Data Set	S	T
BP1	$\begin{pmatrix} 0.8 & 0.2 \\ 0.1 & 0.9 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$
BP2	$\begin{pmatrix} 0.9 & 0.8 & 0.1 \\ 0.1 & 0.9 & 0.8 \\ 0.1 & 0.2 & 0.9 \end{pmatrix}$	$\begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}$
BP3	$\begin{pmatrix} 0.8 & 0.7 & 0.2 & 0.8 \\ 0.9 & 0.3 & 0.8 & 0.2 \\ 0.3 & 0.8 & 0.2 & 0.7 \\ 0.9 & 0.8 & 0.7 & 0.2 \end{pmatrix}$	$\begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$

tribution of offices from 46 global advanced producer service firms over 55 world cities. Global firms are defined as firms owning offices in at least 15 different cities. Service values for a firm in a city are given as 3,2,1 and 0. We binarize the data set such that positive service values become 1 and then generate the bipartite graph. The advanced producer service firms can be categorized into 4 clusters: accountancy firms, advertising firms, banking and finance firms and law firms.

MovieLens² data was collected through the MovieLens web site during the seven-month period from September 19th, 1997 to April 22nd, 1998 by the GroupLens Research Project at the University of Minnesota. It consists of 100,000 ratings (1-5) from 943 users on 1682 movies. We preprocess the data by removing movies which are rated by less than 100 users and users that rated less than 100 movies. Therefore, we got a bipartite graph of 361 users and 334 movies. Then we binarize the preprocessed data set such that 3-5 entries become 1 and others become 0.

Jester³ is a joke rating data set. The original data set contains over 1.7 million continuous ratings (-10.00 to +10.00, +10.00 best) of 150 jokes from 63974 users collected between April 2006 to May 2009. We remove 22 jokes which are never rated or rated by less than 0.01 of users, and randomly pick 1000 users who rate all the picked jokes. Then we binarize the data set such that 5-10 entries become 1 and others become 0. The ground truth is generated for evaluating link prediction, such that the non-negative entries become 1 and the negative entries become 0.

²<http://www.grouplens.org/node/12>

³<http://eigentaste.berkeley.edu/dataset/>

Table 5.2: Clustering Performance on Synthetic Data.

	BP1	BP2	BP3
<i>SCMiner</i>	1	1	0.9949
<i>CA</i>	0.6683	0.7897	0.8750
<i>ITCC</i>	1	1	0.8750
<i>GS</i>	0.2568	0.4069	0.5493

5.4.2 Clustering Quality

Firstly, we evaluate the quality of clusters detected by SCMiner. SCMiner can be used as a parameter-free bipartite graph partition method and therefore we choose the approaches Cross-association (CA) [17] and Information-theoretic Co-clustering (ITCC) [25] as comparison methods. In addition, we compare SCMiner to the graph summarization technique (GS) of [81]. The algorithms SCMiner, CA and GS are both parameter-free, ITCC requires the number of clusters in rows and columns. The algorithm GS does not output a clustering, however, the summarized nodes can also be regarded as clusters. We therefore create a cluster for each summarized node of the algorithm. For synthetic data we set the number of clusters of ITCC to the true number of the data set. We use the Normalized Mutual Information (NMI) [114] to measure the clustering performance. The value of NMI ranges between 0 and 1. The higher the value the better the clustering. We further report the Adjusted Mutual Information (AMI) and Adjusted Variation Information (AVI) scores proposed in [114].

Synthetic Data. Table 5.2 depicts the clustering performance comparison on synthetic data sets evaluated by NMI, which shows that SCMiner yields better results than CA, ITCC and GS. For BP1 and BP2 both ITCC and SCMiner give perfect results, however ITCC needs the true number of clusters as input parameter, whereas SCMiner determines the number of clusters automatically without user input. CA outputs worse NMI results, because it splits the clusters into some smaller ones. Worst NMI results yields GS, this is mainly because GS is designed for summarization but not for clustering. Table 5.2 reports only NMI but the other scores are similar.

For SCMiner we try out several ϵ and choose the best results with the minimum MDL. Figure 5.4 shows the clustering results for all synthetic data sets with different ϵ and one can see that

the results are quite stable on a wide range of ϵ . We also test the relationship between NMI and MDL, for space limitation we only mark several MDL values for synthetic data BP3 in Figure 5.4. The coding costs are 127102 bits when $\epsilon = 0.01$ and $NMI = 0.89$, and it costs 123723 bits when $\epsilon = 0.07$ and $NMI = 0.99$, which proves that smaller MDL values lead to better NMI results.

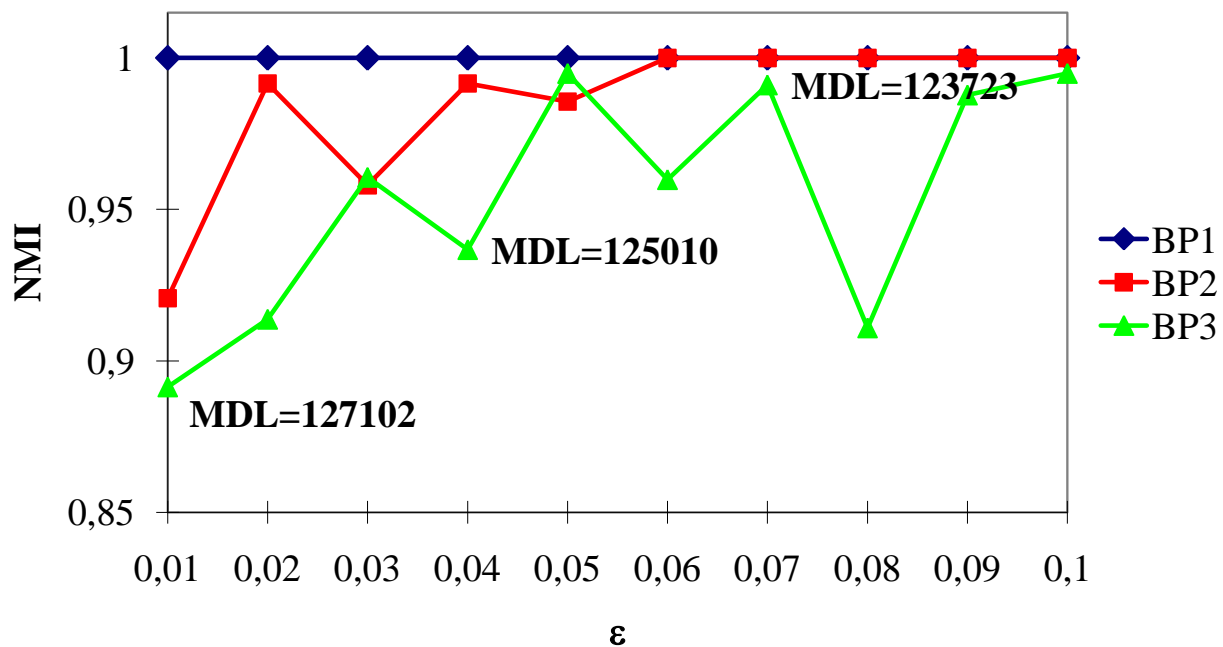


Figure 5.4: Results for Various ϵ .

World Cities Data. The World cities real data set contains cluster labels of global firms and thus we can use NMI to evaluate clustering quality of this type of nodes. We run SCMiner, CA, ITCC and GS on this data set. SCMiner, CA and GS are all parameter-free methods (the chosen ϵ of SCMiner for cities is 0.01). We set the true number of clusters of global firms and 4 as the number of clusters of world cities as input parameter for ITCC. The NMI results for global firms are depicted in Table 5.3 and the table shows that SCMiner clearly outperforms CA, ITCC and GS in all clustering quality scores. The evaluation of cities type clusters is much more difficult, since the cluster labels are not provided along with the data set. Looking in detail at the cluster contents, SCMiner finds 3 clusters and a separated city Washington, DC. The first cluster contains 13 cities, including Atlanta, Boston, Dallas, Munich, Montreal etc. and all these cities

are strong in economics. In detail, all 5 accountancy firms have services in these cities, and there are 3 advertising, 5 banking and 2 law companies in average owning offices in these 13 cities. The second cluster contains 17 cities, including Toronto, Paris, London, New York, and Beijing. All these cities are metropolis in the world. Moreover most of these cities are capital of their country. In terms of service, all 5 accountancy firms have services in these cities, and there are 9 advertising, 11 banking and 7 law companies in average having offices in these 17 cities. The third cluster contains 24 cities, including Amsterdam, Barcelona, Seoul, Shanghai etc., which are all kind of financial cities. In terms of service, all 5 accountancy firms have services in these cities as well, and there are 7 advertising, 8 banking and 2 law companies in average having offices in these 24 cities. The cluster analysis shows that SCMiner outputs reasonable clusters regarding the cities type nodes. To sum up, the cities can be categorized into 3 types, capital metropolis, financial cities and economic cities. Capital metropolis are the economic, financial and politic center of a country, therefore all kinds of companies have offices in these cities. Financial cities offer a lot of banking and advertising company services, but lack law services, because they are not politically oriented. Some local manufactories are located in economic cities, whereas advertising, banking and law companies are not. Washington, DC is quite different compared to the other cities, since it is a politic, but not a financial city and therefore owns lots of law firms' offices but fewer advertising and banking firms' offices. Thus it is reasonable that this city is separated in its own cluster.

Table 5.3: Results on World Cities Data.

	NMI	AMI	AVI
<i>SCMiner</i>	0.4345	0.3807	0.3824
<i>CA</i>	0.3109	0.2447	0.2604
<i>ITCC</i>	0.2522	0.1845	0.1891
<i>GS</i>	0.2515	0.0467	0.0683

MovieLens Data. MovieLens data set does not provide cluster labels, however, it provides movie categories and personal information of users that can be used to analyze the cluster contents. We separately perform SCMiner, CA, ITCC and GS on this data set, SCMiner, CA and GS are both parameter-free (the chosen ϵ of SCMiner for MovieLens is 0.05). CA outputs 9 user

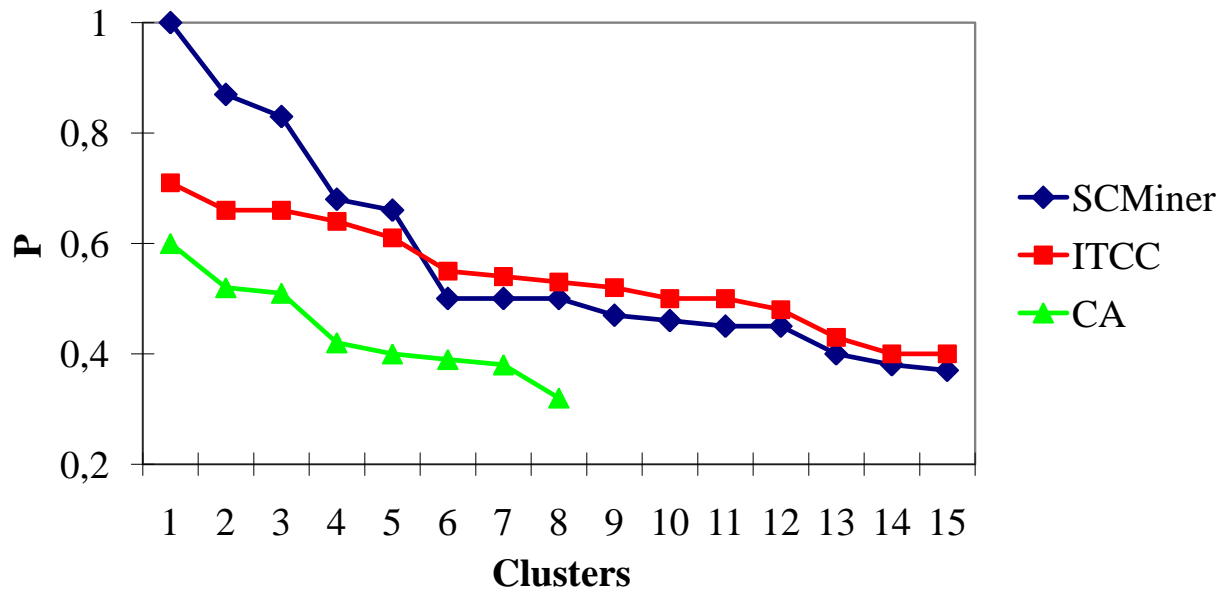


Figure 5.5: Cluster Purity on Movielens Data.

clusters and 8 movie clusters. In addition some isolated nodes and small clusters with less than 5 instances are found. SCMiner gives 10 user clusters and 15 movie clusters, whereas all the clusters found by GS are single nodes or just contain a few nodes (< 5). To be fair, we set the number of user and movie clusters to 10 and 15 for ITCC, which corresponds to the number of clusters found by SCMiner. In this data set, movies are classified into 19 genres, such as action, adventure, animation, etc, and a movie can be categorized into several genres at once. According to the basic concept of clustering that objects in the same cluster are similar, we define purity P to evaluate the movie clusters. By counting genres of movies, we can acquire the genre that dominates the cluster and let this genre be the cluster representative. The purity of a cluster is defined as the percentage of movies belonging to the most represented genre. We calculate the purity P for each movie cluster detected by SCMiner, CA and ITCC, and sort them by p value, which is shown in Figure 5.5. The figure shows that SCMiner outputs more pure movie clusters than the two comparison methods.

In terms of user clusters, it is difficult to compare the results. Analyzing the content of user clusters detected by SCMiner, reveals that some clusters contain old users, some are composed of young users, some groups contain only males, while others own mainly women. This reflects

that SCMiner outputs meaningful user clusters.

5.4.3 Hidden Structure

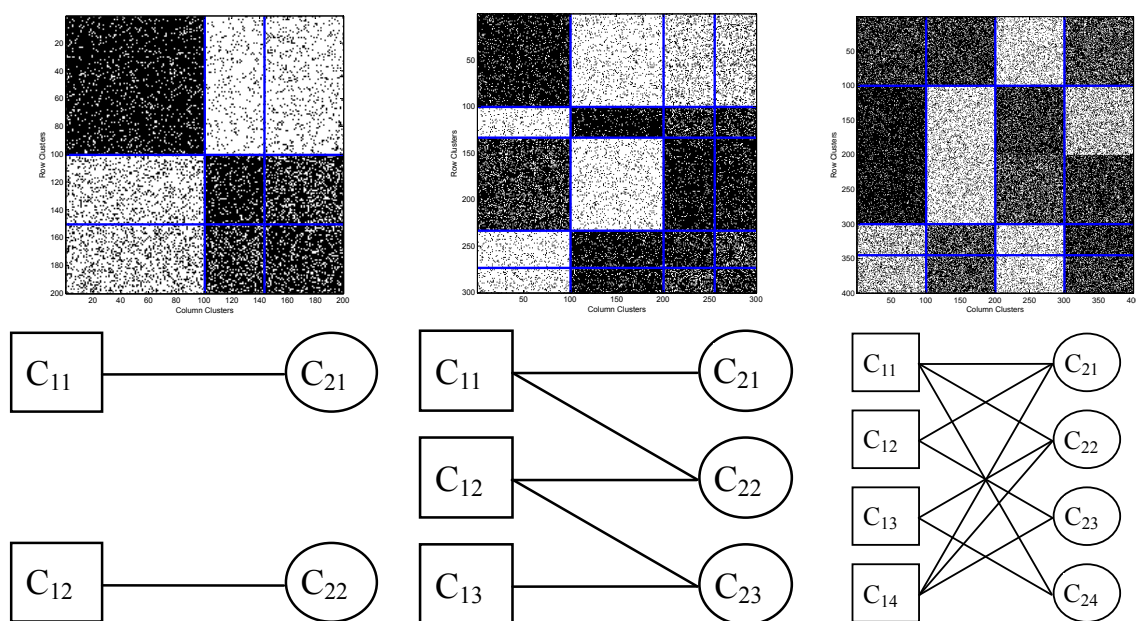


Figure 5.6: Hidden Structure Detected by SCMiner and CA. From Left to Right: Data Set BP1, BP2, BP3.

For bipartite graph data, we do not only want to analyze the clusters in each type, but we also want to evaluate the relationship between clusters of different types. Besides our SCMiner algorithm, CA is the only comparison method providing information about these relationships (in the form of a re-arranged adjacency matrix) to users.

Synthetic Data. Figure 5.6 depicts the hidden structure detected by SCMiner and CA, the top row shows the re-arranged adjacency matrix of data output by CA, the bottom row depicts the hidden structure found by SCMiner, where squares and circles denote the two different types of clusters. For BP1 and BP2 SCMiner gives perfect results and for BP3 just one node is categorized incorrectly, as cluster C_{12} contains 99 nodes and cluster C_{14} contains 101 nodes, but the whole structure is correct, whereas the cross association output by CA for BP3 is incorrect and it is really hard to tell the relationship between the second row and fourth column cluster. In summary,

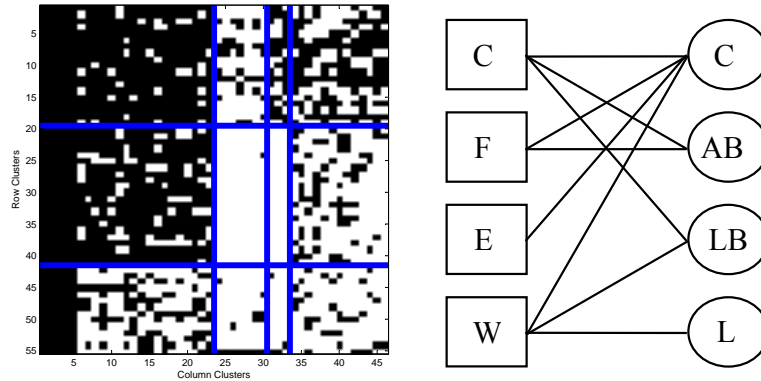


Figure 5.7: Hidden Structure of Cities Detected by CA (left) and SCMiner (right). For SCMiner results, Square C represents a cluster consisting of capitals, Square F financial cities, Square E economic cities and Square W is the isolated city Washington, DC. On the other side, the cluster represented by Circle C mainly contains accountancy firms, Circle AB advertising and banking companies, Circle LB banking and law firms, and Circle L law firms.

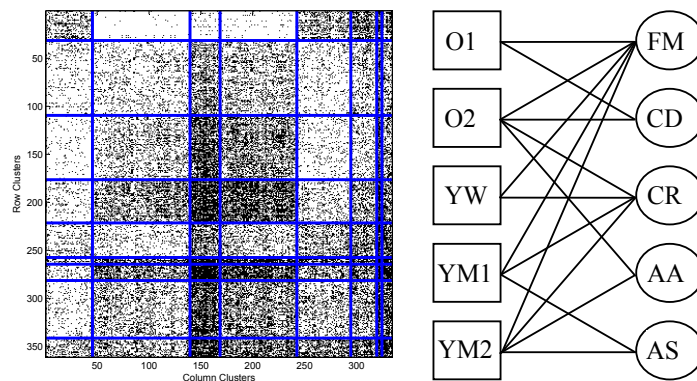


Figure 5.8: Hidden Structure of MovieLens Detected by CA (left) and SCMiner (right). For SCMiner results, Square $O1$ and $O2$ denote clusters containing old users, Square YW represents a cluster of young women, and Square $YM1$ and $YM2$ young man. On the other side, the cluster represented by Circle FM represents high scored movies, Circle CD comedy and drama, Circle CR action, romance and comedy. Circles AA and AS represent adventure, action, thriller movies and action, sci-fi, thriller movies, respectively.

the visualization of Cross-associations is only clear and interpretable for data having a relatively easy structure.

World Cities Data. Figure 5.7 depicts the hidden structure of World cities data set detected by SCMiner and CA. The left shows the re-arranged adjacency matrix of data output by CA. It is hard to identify the relationship between two types of nodes from the matrix itself. The right depicts the hidden structure found by SCMiner, the squares represent the cities clusters and circles denote clusters of companies. The figure shows that capital metropolis have connections to all kind of firms, financial cities do not have law companies services, most service in economic cities are from accountancy companies, and Washington, DC owns lots of law companies but fewer banking and advertising firms. From the figure, the major structure of the complex network becomes obvious at first glance. The result of SCMiner is a highly compact bipartite graph, a data representation which is easy to understand for the user. In contrast, the re-arranged adjacency matrix is still quite noisy and it is very difficult to infer the structure from this representation.

MovieLens Data. Figure 5.8 depicts the hidden structure of MovieLens data set detected by SCMiner and CA, the left shows the re-arranged adjacency matrix of data output by CA, which is hard to understand, the right depicts the hidden structure found by SCMiner which is much easier to analyze. For clarity we only show the relation between the 5 biggest clusters in each type. Squares and circles represent the user and movies cluster. Cluster $O1$ and $O2$ contain user groups that are older than the average regarding the whole data set, and their male-to-female ratio is just about whole data set average. Square YW represents a user group with a much larger female ratio compared to the average, and the users are younger than the average as well. Square $YM1$ and $YM2$ denote clusters with groups of users containing almost all young man. On the other side, circle FM represents a cluster of famous movies, containing Schindler's List, Shawshank Redemption, and Seven etc. that all have high rating scores in IMDB, which shows that all groups of users like it. This fact is embodied in the revealed hidden structure, since this cluster has connections to all user groups. Circle CD is mainly consisting of movies from comedy and drama category, so it makes sense that older people like it. The movies in circle CR are mostly from action, romance and comedy genre, so young women and young men like

it. Circle *AA* and *AS* represent adventure action thriller movies and action sci-fi thriller movies respectively, therefore young man like them. The above analysis shows that the hidden structure found by SCMiner is reasonable.

5.4.4 Link Prediction Accuracy

Table 5.4: Link Prediction Performances.

Algorithm	BP1		BP2		BP3		Jester	
	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall
<i>PA</i>	0.084	0.084	0.436	0.436	0.442	0.443	0.406	0.369
<i>Katz</i>	0.984	0.984	0.943	0.943	0.547	0.548	0.406	0.369
<i>GS</i>	0.965	1	0.981	1	0.994	0.973	0.356	0.05
<i>SCMiner</i>	1	1	1	1	0.997	1	0.431	0.389

SCMiner outputs the summarization of the input graph that yields the minimum coding cost. The summarization itself consists of a summary graph and an additional graph, where the ”-” edges in the additional graph denote missing links which should be added to the original graph, and ”+” edges might be noisy links which should be removed from the original graph. Since it is hard to determine whether a ”-” edge represents noise, we only use link prediction to evaluate whether a ”+” edge is a missing link.

The link prediction problem has been studied on graphs by several approaches which can be divided into two categories, supervised, e.g. [67, 45] and unsupervised methods, e.g. [66]. Since SCMiner is unsupervised, we only compare our results to unsupervised approaches. There are two types of unsupervised methods: based on node neighbors and based on paths between nodes. However, some of these methods, like common neighbor, are not suitable for link prediction on bipartite graph data and therefore we choose two methods, preferential attachment (PA) [75] and Katz [58], that are suitable for bipartite graphs. We use precision and recall to evaluate the accuracy. SCMiner automatically outputs K predicted edges and for reasons of fairness we choose the top K predictions of the ranking list of PA and Katz to compare the precision and recall of the prediction results. We also compare our approach to Graph Summarization (GS), since the algorithm changes the edges of the original graph as well. Table 5.4 shows the link pre-

diction comparison on our synthetic data sets as well as on the real data set Jester. All the results are averaged over runs on 10 randomly generated or the selected data sets. SCMiner predicts the missing links of BP1 and BP2 completely and those of BP3 nearly completely correct and therefore yields better results on all synthetic data sets than all comparison methods. Moreover it outperforms the other approaches on the Jester data set. Interestingly, GS performs better on the synthetic data sets than on Jester. The reason might be that the Jester data set is sparser than the synthetic data sets, and therefore GS reaches a local minimum very fast, which results in fewer predicted edges.

5.5 Related Work and Discussion

During past decades, many algorithms were proposed for graph clustering, graph compression, graph summarization and link prediction.

5.5.1 Co-clustering

Bi-clustering or co-clustering is a creative approach which simultaneously clusters rows and columns of a data matrix. It avoids the problems caused by sparseness and high-dimensionality that traditional one way clustering algorithms suffer from. There are some state-of-the-art co-clustering algorithms, such as Information-theoretic Co-clustering [25], Cross-association [17] etc. Specifically, Information theoretic Co-clustering [25] simultaneously maps row elements to row clusters and column elements to column clusters, mutual information of each clustering state is calculated and compared to the initial state. Thus, the optimal co-clustering result is obtained when the mutual information loss is minimal. However, the drawback of the method is that the number of both row and column clusters must be predetermined. Cross-association [17] is a MDL-based parameter-free co-clustering method that processes a binary matrix and seeks clusters of rows and columns alternately. Then the matrix is divided into homogeneous rectangles which represent underlying structure of the data. However, compared with our algorithm, it only addresses the clustering problem. Moreover, Long et al. [71] proposed a framework

for co-clustering named block value decomposition (BVD), which formulates co-clustering as an optimization problem of matrix decomposition. Similarly, in [70], Long et al. reconstruct a bipartite graph based on some hidden nodes and thus an optimal co-clustering result is obtained from the new bipartite graph which mostly approximates the origin graph. However, these two algorithms both need the number of clusters as input parameter. Besides, Dhillon [23] proposed a spectral algorithm for bipartite graph partition. Shan and Banerjee [96] proposed a Bayesian co-clustering model and considered co-clustering as a generative mixture modeling problem. George and Merugu proposed a co-clustering algorithm based on the collaborative filtering framework [36]. In terms of real life data sets, [20] applied co-clustering on gene expression data and [23] effectively processed documents and words data.

5.5.2 Graph Compression and Summarization

In real life, it is common that a graph consists of tens of thousands nodes and complex linkages. Graph compression and graph summarization [81, 110, 124] are effective ways to simplify the original large graph and to extract useful information. Many algorithms are based on SUBDUE [50], which is a classical graph compression system. It makes use of the MDL Principle to find a subgraph suitable for compression. Bayesian Model Merging (BMM) [101, 102] also is a kind of compression method which is based on Bayesian formula. The best compressed model can be achieved by maximizing the posterior probability. Navlakha et al. [81] represented a graph by a summary graph and a correction matrix, and summarization is implemented by greedy merging and randomized algorithms. MDL is used to find the local optimum. However, the algorithm is designed for compressing only and performs poorly in clustering or link prediction. In [110], according to group nodes based on their attributes and relationships, which are selected by the user, Tian et al. propose a graph summarization algorithm.

5.5.3 Link Prediction

Predicting whether two disconnected nodes will be connected in the future, based on available network information is known as the challenge of link prediction. Liben-Nowell and Kleinberg

[66] summarize some unsupervised link prediction approaches for social networks. For example, common neighbor, Jaccard's coefficient, preferential attachment [75], Katz [58] etc. Specifically, preferential attachment is based on the idea that two nodes with higher degree have higher probability to be connected. Katz [58] defines a score that sums up the number of all paths between two nodes where short paths are weighted stronger. By ranking these scores, the probability of whether two disconnected nodes will be linked in the future can be acquired. However, when dealing with bipartite graphs, Kunegis et al. [63] stated that scores based on common neighbor are not suitable, because two connected nodes do not have any common neighbors in a bipartite graph. But scores like preferential attachment and *Katz*, which are used in this chapter as comparison methods are reasonable.

5.6 Chapter Conclusion

In this chapter, we propose SCMiner a technique for mining knowledge from bipartite graph which integrates graph summarization, bipartite graph clustering, link prediction and hidden structure mining. Based on the sound optimization goal of data compression, our algorithm finds a compact summary representation of a large graph. In addition, while compressing the graph SCMiner detects the truly relevant clusters of both node types and their hidden relationships. Thus, SCMiner is a framework comprehensively supporting the major tasks in unsupervised graph mining. Our experiments have demonstrated that SCMiner outperforms specialized state-of-the-art techniques for co-clustering and link prediction. In ongoing and future work we want to extend this idea to support multi-partite graphs as well as graphs with different edge types.

Chapter 6

Detection of Overlapping Communities in Attributed Graphs

As the information contained in the data is becoming more complex, the complexity of the graph generated from the data is increasing. In this chapter, we focus on mining knowledge from the attributed graph with both structural and attribute information. To be specific, the structural information conveys the relationship between each pair of objects, and the attribute information expresses features of each object. Therefore, the results of mining attributed graph are decided by both aspect of information. In this chapter we propose an algorithm called IROC (for **I**nformation-theoretic non-**R**edundant **O**verlapping **C**lustering). IROC improves the clustering results from many aspects. Firstly, IROC aims at discovering overlapping clusters which means nodes as well as attributes are allowed to be assigned to multiple different clusters. Moreover, the overlapping clusters detected by IROC avoid the redundancy issue. After executing IROC, besides the non-redundant overlapping clusters we can also obtain the attribute subspace of each cluster which expresses the common meaning of the cluster. In addition, IROC is proposed based on Minimum Description Length principle, thus it does not require any input parameters.

The remainder of this chapter is organized as follows: Section 6.1 gives the introduction. Section 6.2 describes the coding scheme. Section 6.3 presents the algorithm in detail. Section 6.4 contains an extensive experimental evaluation. Section 6.5 discusses related work and Section

6.6 gives the conclusion of this chapter.

Parts of the material presented in this chapter have been submitted to [31], where Jing Feng was mostly responsible for the development of the main concept, implemented the main algorithms and wrote the largest parts of the paper; Nina Hubig wrote parts of the paper and revised the whole paper; Xiao He performed part of experiments; Claudia Plant supervised the project and revised the whole paper; The co-authors also contributed to the conceptual development and paper writing.

“Jing Feng, Nina Hubig, Xiao He, Claudia Plant. Detection of Overlapping Communities in Attributed Graphs. Submitted for publication.”

6.1 Introduction

6.1.1 Motivation

Social networks, gene and/or protein interaction networks as well as nearly all common types of networks in real-world applications are not only sharing a large amount of information depending on the relationship between the vertices, but also contributing information regarding the characteristics of these vertices. These characteristics are modeled as attributes of a vertex. Thus we are referring to a graph containing extra information in its nodes as an *attributed graph*. These attributed graphs connect two aspects of information: First, the structural “who knows whom” given by the graph itself as for example in friendship relationships of applications like Facebook. Second, the attributes per node or person showing (in the case of Facebook) personal information like hobbies, what they are working, where they are living etc. Combining both aspects of information gives the chance to answer questions like “what this friendship circle has in common?” or “what do most people in my work environment like and what joins them together?”. Questions like this can be answered to some extent by just clustering attributed graphs, as existing algorithms propose in [128, 119, 127].

But we are not only clustering attributed graphs but are also applying an overlapping concept to both informational aspects to the graph structure and to the attribute space. What do we mean

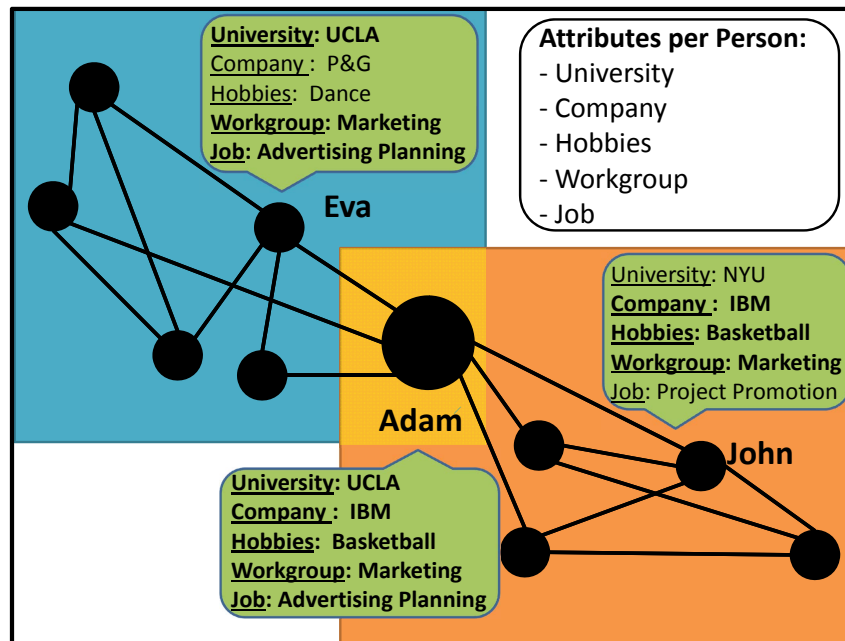


Figure 6.1: Motivational Example of a Social Friendship Network.

by overlapping and what do we gain out of the attributed graph if we are already able to group both informational aspects by just partitioning the attributed graph?

Consider the example given in Figure 6.1, it outlines the meaning of “overlapping” in combination with an attributed graph. The figure shows the friendship circles in a social network of a person called “Adam”. His two friendship circles are visualized in two rectangles behind the graph structure: the top left stands for his friends from school and the bottom right indicates his circle of colleagues. The overlapping of both circles is shadowed. Every person (node) includes attributes on their university, their working place and department, what hobbies they have and what job position they are in. You could also think of these attributes as more of a list, containing several items and differing in detail. For example, Adam graduated from UCLA, works at IBM, likes playing basketball, works in the marketing department and is mainly in charge of planning advertisements. Now, how would an attributed graph partitioner most likely divide this structure if no overlap is allowed? A distinct possibility is to assign Adam to either the school friends or the colleagues circle, because Adam can form a nearly full clique with both circles. This would result in a very high quality for clustering for one cluster while the other would lose some infor-

mation (Adam). In this frequent case, partitioning must sacrifice the quality of this one cluster. Obviously, overlapping clustering, like assigning Adam to both circles, provides more meaningful structural information. Lets take a look at the two named nodes John and Eva that share some overlap in their attributes with Adam. John is a working partner of Adam, and shares the same hobby, workgroup and of course company with him. Also Eva graduated from the same university, shares the same workgroup and is working in a similar job like Adam, just in a different company. Therefore, Adam shares similar attributes with his work circle where John is, while he shares other attributes with his school circle where Eva is. Apparently Adam should be assigned to both circles from the attribute information side as well. All in all this small example already shows the realistic significance when overlapping is integrated in structure and attribute space. However, in an extreme example, overlapping can appear as the whole smaller cluster is included in a bigger cluster. Such overlapping will cause *information redundancy*, which basically means that the smaller cluster includes no further interesting information. In this chapter, the proposed method aims to detect overlapping clusters and meanwhile avoid information redundancy.

6.1.2 Contributions

Therefore, we contribute a new method of clustering attributed graphs with the objective to find the reasonable overlapping communities and meaningful subspace of the attributes at the same time based on an information theoretical technique. The advantages of the proposed algorithm are in short:

- **Discovery of overlapping communities:** Our method discovers not only single dense groups of friends but also their connections to other groups. Node structures can have several group memberships and be efficiently evaluated.
- **Finding coherent attribute subspaces:** We think that some information is hidden in attribute subspaces that can be obtained to express the meaning of the cluster. Overlapping is also allowed among attribute subspaces.
- **No Redundancy:** Based on the Minimum Description Length (MDL) principle [92], our

approach balances quality and redundancy in both graph structure and attribute space. Specifically, a node is only assigned to several graph clusters and an attribute is only part of multiple subspaces if this pays-off in terms of data compression. Thereby, only the truly relevant overlap useful to compress the data is reported as a result of our algorithm.

- **Automation:** Our information theoretic based algorithm relieves the user from the task of finding parameters to run the method on every specific data. The non-redundant overlapping clusters and coherent attribute subspace can be detected automatically.

6.2 Compressing an Attributed Graph

To understand our coding scheme it is foremost important to outline what needs to be compressed in an attributed graph. From start, an attributed graph is an extension from a general graph by involving attributed information to each vertex. Therefore, two types of matrices are needed to model both structural connections in the graph and the attribute space of each node. Same as the general graph, the structure is mapped into an adjacency matrix, while the attributes of each vertex can be modeled as a matrix with rows denoting vertices and columns representing attributes. We name this matrix the *attribute matrix*. So far, an attributed graph is represented by an adjacency matrix and an attribute matrix that need to be compressed for efficiency and automation. For simplicity in this chapter, we focus on undirected unweighed graphs with categorical attributes.

6.2.1 Notations

Before we start with the coding scheme, this section describes the notation and used throughout the chapter.

Definition 6.1 (Attributed Graph) *An attributed graph is defined as $G = (V, E, \Lambda)$, where $V = \{v_1, v_2, \dots, v_{|V|}\}$ contains $|V|$ vertices, $E = \{(v_i, v_j), 1 \leq i \leq |V|, 1 \leq j \leq |V|, i \neq j\}$ are edges, $\Lambda = \{\lambda_1, \lambda_2, \dots, \lambda_T\}$ are T attributes of each vertex. $\Theta = \{\theta_1, \theta_2, \dots, \theta_T\}$ represents domains of Λ . And θ_k is all possible values of attribute λ_k , where $k \in \{1, 2, \dots, T\}$.*

Besides, \mathcal{A} is the adjacency matrix with $\mathcal{A}_{i,j} = 1$ if $(v_i, v_j) \in E$ and \mathcal{F} is the attribute matrix with $\mathcal{F}_{i,k}$ denoting the categorical value of the i th vertex in the k th attribute. An attribute graph is represented as $G = (\mathcal{A}, \mathcal{F})$ as well.

In this chapter, we aim at mining knowledge from the attributed graph by detecting non-redundant overlapping clusters. As attributed graphs possess both structural and attribute information, the cluster of such type of data covers both information, which is defined in Definition 6.2. A cluster C is a subset of an attributed graph G . Specifically, the cluster needs to be densely connected and contains a subgroup of attributes to describe the meaning of the cluster.

Definition 6.2 (Cluster in an Attributed Graph) A cluster is defined as $C = (V', E', \Lambda')$ that is a subset of the attributed graph G , where $V' \subseteq V$, $E' \subseteq E$, $\Lambda' \subseteq \Lambda$. $V' = \{v'_1, v'_2, \dots, v'_{|V'|}\}$ is a subset of V that includes $|V'|$ densely connected vertices and $\Lambda' = \{\lambda'_1, \lambda'_2, \dots, \lambda'_S\}$ is a subset of Λ that contains S attributes with coherent categories, where $S \leq T$.

$\mathcal{A}_C \subset \mathcal{A}$ is the subset adjacency matrix of the cluster C that only contains the vertices in C . And $\mathcal{F}_C \subset \mathcal{F}$ is the subset attribute matrix of the cluster C . Similarly, a cluster in an attributed graph can be represented as $C = (\mathcal{A}_C, \mathcal{F}_C)$ as well.

Besides the own structure of clusters, there are some edges connecting these clusters that are not included inside the clustering. Similarly to the edge structure, many attributes from the full-dimensional subspace are not assigned to any cluster. We define these areas as the non-cluster area of an attributed graph in Definition 6.3, which consists of the elements lying outside any cluster in both structure and attribute space.

Definition 6.3 (Non-Cluster Area in an Attributed Graph) The non-cluster area of an attributed graph G modeled by K clusters $\{C_1, C_2, \dots, C_K\}$ is defined as $U = U_{\mathcal{A}} \cup U_{\mathcal{F}}$, where $U_{\mathcal{A}}$ is the non-cluster area in the adjacency matrix \mathcal{A} and $U_{\mathcal{F}}$ is the non-cluster area in the attribute matrix \mathcal{F} . $U_{\mathcal{A}} = \mathcal{A} \setminus \mathcal{A}'$, where $\mathcal{A}' = \mathcal{A}_{C_1} \cup \mathcal{A}_{C_2} \dots \cup \mathcal{A}_{C_K}$ is the combination of all structural elements appearing in $\{C_1, C_2, \dots, C_K\}$ and $U_{\mathcal{F}} = U_{\lambda_1} \cup U_{\lambda_2} \dots \cup U_{\lambda_T}$, where U_{λ_k} with $k \in \{1, 2, \dots, T\}$ are entries of \mathcal{F} in attribute λ_k which are not included in $\{C_1, C_2, \dots, C_K\}$.

6.2.2 Coding Scheme

The coding scheme is the core part of our approach and is divided into three parts: first the basic ways of how compression is used in information theory for graphs, and then how the specific data costs for an attributed graph are calculated as well as how a specific data structure like that can be modeled.

As a lossless compression, Minimum Description Length (MDL) [92] principle follows the assumption that the less coding length we adopt to describe the data, the more knowledge we can gain from it. Formally, the quality of a model can be identified from Eq.(6.1), where $L(M)$ denotes the coding length for describing model M and its parameters, while $L(D | M)$ represents the cost of coding data D under model M .

$$L(M, D) = L(D | M) + L(M) \quad (6.1)$$

In our cases, D is the attributed graph G . The length of the data under the model $L(D | M)$ can be described as the coding cost of clusters and non-cluster area in the attributed graph G . The length of the model $L(M)$ is the coding length of assignment of both vertices and attributes and the coding cost of parameters. In the following, we elaborate the data description and the model costs that are necessary to compress an attributed graph in detail.

Data Description Cost $L(D | M)$

Suppose K clusters $\{C_1, C_2, \dots, C_K\}$ are discovered from an attributed graph G . Under the clustering model, the attributed graph can be described as K clusters $\{C_1, C_2, \dots, C_K\}$ and a non-cluster area U . Therefore the data description cost $L(D | M)$ is equivalent to all costs of all clusters plus the non cluster area, as shown in Eq.(6.2). In the following, we will describe the data description cost in specific.

$$L(D | M) = \sum_{i=1}^K CC(C_i) + CC(U) \quad (6.2)$$

Coding cost of a cluster $CC(C_i)$: A cluster C_i can be represented by the subset adjacency matrix \mathcal{A}_{C_i} and the subset attribute matrix \mathcal{F}_{C_i} . Then the coding cost of the cluster C_i is the sum of the structural coding cost $CC^A(C_i)$ and the attribute coding cost $CC^F(C_i)$, as shown in Eq. (6.3).

$$CC(C_i) = CC^A(C_i) + CC^F(C_i) \quad (6.3)$$

In structural aspect, cluster C_i is composed of densely connected vertices which equals to high probability ‘1’ in subset adjacency matrix \mathcal{A}_{C_i} . The average coding cost of the entries in matrix \mathcal{A}_{C_i} is lower bounded by its entropy. Because we consider G as an undirected graph, we only need to encode the entries of the upper triangular matrix. Therefore, the coding cost of the structural information of the cluster C_i is described in Eq. (6.4), where $p_1(C_i)$ and $p_0(C_i)$ stand for the probability of ‘1’ and ‘0’ in the subset adjacency matrix \mathcal{A}_{C_i} respectively. And n_{C_i} refers to the number of entries in upper triangular matrix of \mathcal{A}_{C_i} .

$$CC^A(C_i) = -n_{C_i} \cdot (p_1(C_i) \cdot \log_2 p_1(C_i) + p_0(C_i) \cdot \log_2 p_0(C_i)) \quad (6.4)$$

In attribute aspect, the subspace of the cluster C_i is described as a subset attribute matrix \mathcal{F}_{C_i} with the size $N_{C_i} \times S$. S attributes are chosen from T attributes to represent the meaning of the N_{C_i} densely connected vertices. Thus each attribute in subspace need to be generally identical. How to encode the subset attribute matrix \mathcal{F}_{C_i} ? Figure 6.2 depicts a codebook which is adopted to encode the attribute information of cluster C_i . As shown in the figure, attributes of a vertex in cluster C_i can be represented as a category string with the length equals to the subspace. The codebook shows R groups of category strings of cluster C_i and vertices in each group contain identical categories. The probabilities of each group are given as $p_{g1}, p_{g2}, \dots, p_{gR}$. Additionally, we use $\log_2 n_{\theta_k}$ bits to encode each category string, and n_{θ_k} is the number of categories in attribute k included in subspace S . Then the coding cost of the attribute information of cluster C_i

can be calculated by Eq. (6.5).

$$CC^{\mathcal{F}}(C_i) = -R \cdot \sum_{r=1}^R p_{gr} \cdot \log_2 p_{gr} + R \cdot \sum_{k=1}^S \log_2 n_{\theta_k} \quad (6.5)$$

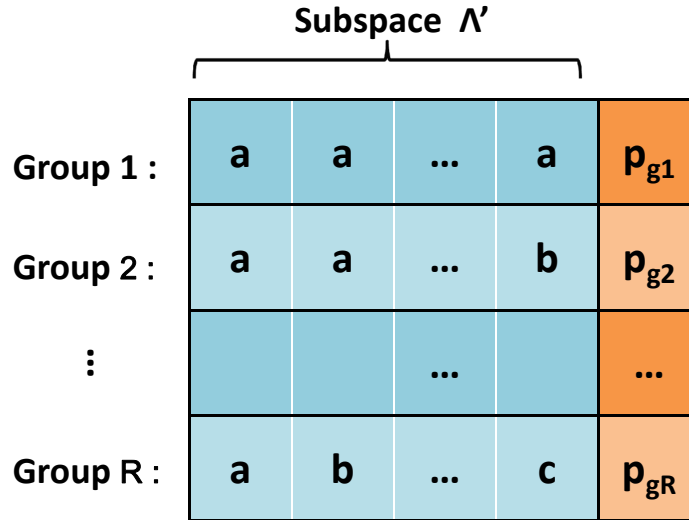


Figure 6.2: The Codebook of an Attribute Matrix

Coding cost of the non-cluster area $CC(U)$: The non-cluster area consists of the elements in the adjacency matrix \mathcal{A} and the attribute matrix \mathcal{F} that are not contained in K clusters, which are represented as $U_{\mathcal{A}}$ and $U_{\mathcal{F}}$ respectively. Similarly, the coding cost of the non-cluster area $CC(U)$ is the sum of the structural coding cost $CC(U_{\mathcal{A}})$ and the attributed coding cost $CC(U_{\mathcal{F}})$, as shown in Eq. (6.6).

$$CC(U) = CC(U_{\mathcal{A}}) + CC(U_{\mathcal{F}}) \quad (6.6)$$

We consider all elements of the structural non-cluster area $U_{\mathcal{A}}$ entirely and code them with a fixed coding order. The coding cost of the non-cluster area of the structural aspect $CC(U_{\mathcal{A}})$ can be encoded as shown Eq. (6.7). Here, $p_1(U_{\mathcal{A}})$ is the number of edges in $U_{\mathcal{A}}$ and $p_0(U_{\mathcal{A}})$ is the number of non-edges in $U_{\mathcal{A}}$. Due to the symmetry property of the matrix, $n_{U_{\mathcal{A}}}$ equals to half of

the elements in $U_{\mathcal{A}}$.

$$CC(U_{\mathcal{A}}) = -n_{U_{\mathcal{A}}} \cdot (p_1(U_{\mathcal{A}}) \cdot \log_2 p_1(U_{\mathcal{A}}) + p_0(U_{\mathcal{A}}) \cdot \log_2 p_0(U_{\mathcal{A}})). \quad (6.7)$$

In the non-cluster area of attribute aspect $U_{\mathcal{F}}$, we encode the remaining categories of every attribute in Λ one by one. The coding cost $CC(U_{\mathcal{F}})$ is calculated by Eq. (6.8), where N_{λ_k} is the number of categories of attribute λ_k that are not assigned to any clusters, and p_{θ_j} is the probability of an category θ_j in the remaining elements of each attribute.

$$CC(U_{\mathcal{F}}) = - \sum_{k=1}^T \sum_{j=1}^{\theta_j} N_{\lambda_k} \cdot p_{\theta_j} \cdot \log_2 p_{\theta_j}. \quad (6.8)$$

Model Cost $L(M)$

For encoding the model cost $L(M)$ of the attributed graph, each cluster will be compressed in three aspects: the assignments of each vertex, the assignments of each attribute and the parameters of the clusters. Therefore, the model cost can be calculated by Eq. (6.9).

$$L(M) = \sum_{i=1}^K (CC_{IDV}(C_i) + CC_{IDF}(C_i)) + CC_{para}. \quad (6.9)$$

Coding cost of vertices assignment $CC_{IDV}(C_i)$: As overlapping is allowed in our proposed algorithm, a vertex can be assigned to multiple clusters. For each cluster, we adopt an assignment list L_V with the length of $|V|$ to label the existence of the vertices, which is shown in Figure 6.3. When the vertex belongs to the cluster, the corresponding value in the list is set to 1, otherwise is set to 0. Therefore, the coding cost of the vertices assignment for a cluster $CC_{IDV}(C_i)$ is lower bounded by its entropy as shown in Eq. (6.10), where $p_1(L_V)$ and $p_0(L_V)$ denote the probability of ‘1’ and ‘0’ in the assignment list L_V of cluster C_i respectively.

$$CC_{IDV}(C_i) = -|V| \cdot (p_1(L_V) \cdot \log_2 p_1(L_V) + p_0(L_V) \cdot \log_2 p_0(L_V)). \quad (6.10)$$

Coding cost of attributes assignments $CC_{IDF}(C_i)$: In our proposed algorithm, the corre-

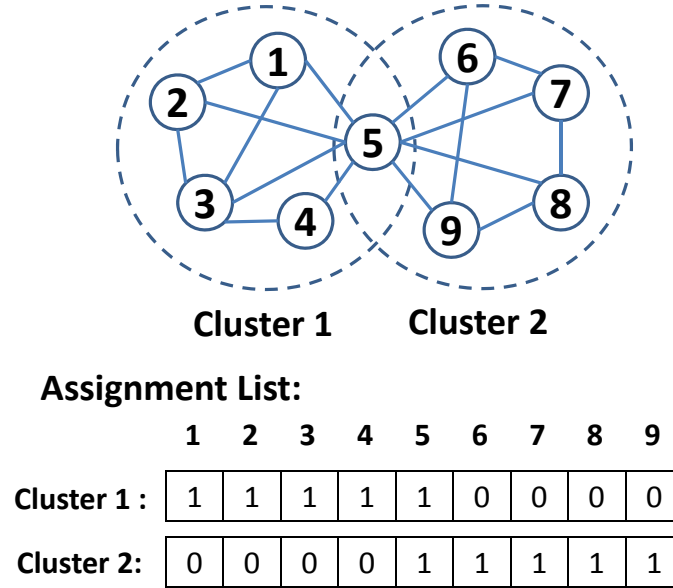


Figure 6.3: The Assignment List of Vertices

sponding attribute subspace of each cluster is also detected. There is overlapping among these attribute subspaces as well. Here we also adopt an assignment list L_Λ with the length of T to represent that attributes are contained in a subspace or not. The coding cost of the attribute assignment of a cluster C_i can be calculate by Eq. (6.11), where $p_1(L_\Lambda)$ and $p_0(L_\Lambda)$ denote the probability of ‘1’ and ‘0’ in the assignment list L_V of cluster C_i respectively..

$$CC_{IDF}(C_i) = -T \cdot (p_1(L_\Lambda) \cdot \log_2 p_1(L_\Lambda) + p_0(L_\Lambda) \cdot \log_2 p_0(L_\Lambda)). \quad (6.11)$$

Coding cost of parameters: To ensure that the receiver acquires the complete information, all parameters need to be encoded to keep the compression lossless. The cost of these parameters can be calculated by Eq.(6.12), where n_p is the number of parameter and n_E is the number of entries. In the following, we will introduce the parameters we need to coding in the proposed algorithm.

$$CC_{para} = 0.5 \cdot n_p \cdot \log_2 n_E. \quad (6.12)$$

Firstly, we need to encode the probability of ‘1’ and ‘0’ in each cluster C_i and the non-cluster area

U . For each cluster C_i , $n_p = 1$ and n_E equals to the half the number of the entries of the subset adjacency matrix \mathcal{A}_{C_i} . Secondly, we encode the probabilities of the R groups in the codebook. Here $n_p = R - 1$ and n_E equal to the number of categories of the codebook. Thirdly, we encode the probabilities of all categories in each attribute in the non-cluster area of the attribute matrix. For each attribute k , $n_p = \theta_j - 1$ and n_E equals to the size of the remaining categories of attribute k . Finally, we encode the probabilities of the assignment lists as parameters as well. For each assignment list, $n_p = 1$ and n_E equals to the size of the list.

6.3 Algorithm IROC

In this section, we propose the algorithm IROC to heuristically efficiently detect overlapping clusters in attributed graphs. The initialization phase and the refinement phase are the two key steps in IROC. The initialization phase is again divided into two subroutines for a) creating graph substructures and b) finding their coherent attribute subspace. The refinement phase takes the responsibility of improving the quality of initial clusters by a) removing redundant parts of the cluster and b) reassigning vertices between two clusters.

6.3.1 Initialization

Creating Graph Substructures: First of all, we create some small subgraphs. Each of them is simply composed of a vertex and all its neighbors. This is an intuitive way to obtain all these small subgraphs. Therefore, $|V|$ subgraphs $SS = \{ss_1, ss_2, \dots, ss_{|V|}\}$ with overlapping are extracted from an attributed graph G , where $|V|$ is the number of vertices. These $|V|$ subgraphs are overlapping with each other, but some of the overlapping parts are reduplicate which produce over much redundancy information. In order to eliminate such over much redundancies, we select K_s subgraphs as initial rough clusters based on the MDL principle automatically. And the Algorithm 6.1 shows the procedure of selecting the K_s clusters $C = \{C_1, C_2, \dots, C_{K_s}\}$ from the $|V|$ subgraphs. The subgraph added to clusters C which reduces the total coding cost most will be selected. To be specific, we consider $|V|$ subgraphs as rough clusters. And we suppose

that there is no cluster in the attributed graph G in the beginning, $C = \emptyset$. Then we add each rough subgraph SS to clusters C and calculate the coding cost by Eq. (6.1) separately. After that we select the subgraph with the minimum coding cost as the first cluster. We keep the selected cluster in clusters C . Analogously, in the remaining $|V| - 1$ candidates subgraphs, we try to add each candidate as the second cluster and choose the one with the minimum coding cost repetitively. This process stops until all vertices are processed, so that K_s initial clusters are chosen automatically.

Finding a Coherent Attribute Subspace: After obtaining K_s rough clusters, we also need to find out the attribute subspace of each rough clusters. Algorithm 6.2 demonstrates the progress of searching an attribute subspace $\Lambda' = \{\lambda'_1, \lambda'_2, \dots, \lambda'_S\}$ of a cluster C_i based on the MDL principle, where S is the size of the subspace. Every vertex of a rough cluster possesses a category in each attribute. For each cluster, we calculate the entropy of each attribute to measure the purity of the categories in the attribute. Entropy in this case refers to the consistence of categories of an attribute. The lower the value is, the higher the consistency of the attribute will be. Thus we order the attributes in ascending order based on their entropies. If some attributes possess the same value, they are formed into groups and considered as one candidate. Thus the candidates of attributes are $\Lambda^o = \{\Lambda_1^o, \Lambda_2^o, \dots, \Lambda_{T_g}^o\}$. Based on the prerequisite that the subspace of the cluster is initialized with an empty set, $\Lambda' = \emptyset$, we add the ordered attribute candidates to the subspace one by one, and meanwhile calculate the coding cost of the whole graph G by Eq. (6.1). Then we select the attribute subspace Λ' of a cluster C_i with the minimum coding cost. In the same way, the attribute subspace of other $K_s - 1$ clusters are chosen.

6.3.2 Refinement

Removing Redundancy: After the initialization step, we receive K_s small attributed clusters which contain *redundant* information. These redundancies serve to generate and find the overlapping in the beginning but are later unwelcome. Therefore our heuristic bottom up algorithm needs to merge these small initial rough clusters to remove redundancies. We define an information-theoretic based similarity which measures the degree of the redundancy between every pair of

Algorithm 6.1 Creating Subgraphs

Input: Attributed Graph G with $|V|$ Vertices and T Attributes**Output:** K_s Rough Clusters: $C = \{C_1, C_2, \dots, C_{K_s}\}$

- 1: Construct $|V|$ subgraphs $SS = \{ss_1, ss_2, \dots, ss_{|V|}\}$ as initial $|V|$ clusters;
 - 2: $C \leftarrow \emptyset$;
 - 3: **while** All vertices are selected **do**
 - 4: **for** All substructures in SS **do**
 - 5: Calculate coding cost of $ss_i \cup C$, $ss_i \in SS$;
 - 6: **end for**
 - 7: Select ss_m with minimum coding cost, $ss_m \cup C$;
 - 8: Remove ss_m from SS ;
 - 9: **end while**
 - 10: **return** K_s Rough Clusters: $C = \{C_1, C_2, \dots, C_{K_s}\}$.
-

Algorithm 6.2 Finding Subspace

Input: A Cluster C_i with $|V'|$ Vertices and T Attributes**Output:** Subspace $\Lambda' = \{\lambda'_1, \lambda'_2, \dots, \lambda'_S\}$

- 1: $\Lambda' \leftarrow \emptyset$;
 - 2: **for** i from 1 to T **do**
 - 3: Calculate entropy of each attribute λ_i ;
 - 4: **end for**
 - 5: Group attributes with same entropy and arrange attributes in ascending order $\Lambda^o = \{\Lambda_1^o, \Lambda_2^o, \dots, \Lambda_{T_g}^o\}$;
 - 6: **for** i from 1 to T_g **do**
 - 7: $\Lambda_i^o \cup \Lambda'$, $\Lambda_i^o \in \Lambda^o$;
 - 8: Calculate coding cost CC of all clusters;
 - 9: **if** CC increases **then**
 - 10: break;
 - 11: **end if**
 - 12: **end for**
 - 13: **return** Subspace $\Lambda' = \{\lambda'_1, \lambda'_2, \dots, \lambda'_S\}$.
-

clusters. Due to the fact that both structural and attribute information are present, we measure the similarity of two clusters as shown in Eq. (6.13):

$$Sim(C_i, C_j) = H(O_A) + H(O_F). \quad (6.13)$$

where $H(\cdot)$ is entropy, O_A and O_F are the structural and attribute overlapping part of cluster C_i and C_j respectively. This quality function tackles the chief problem of merging two clusters: the amount of overlaps in structural and attribute information. How can two clusters be merged without creating new redundancy if they are overlapping in both structure and attribute spaces? On the structural aspect, entropy pays off in the denser overlapping area. Also the smaller the entropy is, the denser the overlapping part of the cluster will be. Thus the smaller entropy of structural overlapping part leads us to detect dense clusters. In contrast on the attribute side, entropy measures the purity of categories in the overlapping area of the attributes. This ensures that the cluster is provided with same categories in the attributes. The smaller the entropy is, the higher the similarity of the attributes in the overlapping part will be. Thus the smaller entropy of attribute overlapping part leads us to detect clusters with coherent meaning. Therefore, the defined similarity balances the entropy of the two aspects, which guides us to merge the two most similar clusters. And in every search run we merge the pair of clusters that share a minimal similarity.

Assigning Vertices: Obviously, merging two clusters C_i and C_j to form a new cluster C_{new} is able to remove redundancy. However, such merging may not eliminate all redundancies. Depending on how accurately the removing of the redundancy works, we need to modify the cluster further in a second refinement step which is shown in Algorithm 6.3. For all vertices $\{v_1, v_2, \dots, v_{N_{C_{new}}}\}$ in C_{new} , we try to split a vertex v_1 from C_{new} and consider it as a new cluster $C_{split} = \{v_1\}$. Then we find the subspace of C_{split} and refine the subspace of $C_{new} = \{v_2, \dots, v_{N_{C_{new}}}\}$. If such process reduces the coding cost, we keep the modification and try to move the next vertex v_2 from C_{new} to C_{split} . If the coding cost is not reduced, we move the vertex v_1 back to C_{new} . The refinement of cluster C_{new} ends when the coding cost achieves its local minimum. If cluster C_{split} is not empty, we treat C_{split} as a new candidate and add it to

the clusters.

Algorithm 6.3 Assigning Vertices

Input: Cluster C_{new}

Output: Cluster C_{new} and Cluster C_{split}

- 1: Create a cluster C_{split} , $C_{split} \leftarrow \emptyset$;
 - 2: Calculate coding cost of all the clusters CC ;
 - 3: **for** All nodes in C_{new} **do**
 - 4: $v \in C_{new}$, remove v from C_{new} , add v to C_{split} ;
 - 5: **Finding Subspace** of C_{split} and C_{new} ;
 - 6: Calculate coding cost of all the clusters CC_{new}
 - 7: **if** $CC_{new} > CC$ **then**
 - 8: Restore v to C_{new} ;
 - 9: **end if**
 - 10: **end for**
 - 11: **return** C_{new} and C_{split} .
-

6.3.3 Overall procedure

The overall procedure of IROC is shown in Algorithm 6.4. First, we automatically select K_s rough clusters and search their attribute subspace as described in the initialization phase. Then we calculate the similarity of every pair of clusters, and merge the two cluster with a minimum similarity. After that a new cluster C_{new} is formed and we find the subspace of it. Then we try to assign vertices from C_{new} to C_{split} under the control of MDL. And we consider C_{split} as a new cluster and recalculate the similarity of every pair of clusters. The merging process continues iteratively and is ended when the coding cost of all clusters achieve its local minimum. Finally, K clusters with coherent attribute subspaces without redundancy are output.

Suppose we have an attributed graph G with $|V|$ vertices, $|E|$ edges and T attributes for each vertex. We first analyze the complexity of the coding when we have K clusters. For each clusters we need to count the edges of the cluster and go through its vertices and attributes to get the probability distributions. The complexity of these processes for all clusters are $O((K + 1) \cdot (A_{vE} + A_{vN} \cdot A_{vT}))$, where $A_{vE} < |E|$ is the average number of edges, $A_{vN} < |V|$ is the average number of vertices and $A_{vT} < T$ is the average number of subspace attributes. The multiplier

Algorithm 6.4 IROC**Input:** Attributed Graph G **Output:** K Clusters with Subspace $C = C_1, C_2, \dots, C_K$

- 1: **Creating Subgraphs** $C = C_1, C_2, \dots, C_{K_s}$;
- 2: **for** i from 1 to K_s **do**
- 3: **Finding Subspace** of C_i
- 4: **end for**
- 5: **while** Not Converge **do**
- 6: Calculate similarity of every pair of clusters;
- 7: Merge two most similar clusters as C_{new} ;
- 8: **Finding Subspace** of C_{new} ;
- 9: **Assigning Vertices** of C_{new} ;
- 10: **end while**
- 11: **return** $C = C_1, C_2, \dots, C_K$.

factor is $K + 1$ considering the no-cluster area. Since the number of cluster is $K \ll |V|$, the complexity of the proposed coding scheme is $O(|E| + |V| \cdot T)$.

In initialization phase of IROC, we greedily choose the initial clusters, the complexity is $O(|V| \cdot |E|)$, where K_s is the number of initial clusters. Then we need to find the subspace for each cluster. Its complexity is $O(K_s \cdot |V| \cdot T^2)$. The reason is that for each cluster we need to calculate the coding cost T times. Finally, the complexity of the full initialization phase (both parts combined) is $O(K_s \cdot |V|(|E| + T^2))$.

In the full refinement step, we need to process K_s^2 pairs of clusters. In each merging and splitting process, we need to move vertices to other clusters and then calculate the subspace of the new clusters. Therefore, we need $O(|V|^2 \cdot T^2)$ time to do so. Finally, the complexity in this step is $O(K_s^2 \cdot |V|^2 \cdot T^2)$.

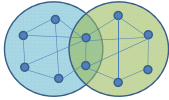

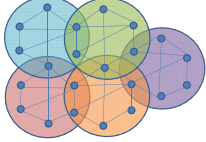
6.4 Experiments

In this section, we evaluate our proposed algorithm IROC on both synthetic and real data sets and compare it with 3 relevant algorithms. The code of the comparison methods are obtained from the authors. Experiments have been performed on a workstation with 2.9GHz Intel Core i7 and 8G memory.

6.4.1 Synthetic Data sets

Data sets. The sketches of three synthetic data sets are shown in Table 6.1. Each circle represents a densely connected cluster and the shadow part denotes the overlapping part in which vertices are assigned to multiple clusters. The synthetic attributed graphs are generated with the following parameters: Each cluster contains 200 vertices ($n_c = 200$); n_o is the number of vertices in the overlapping part; the density of edges in and between clusters are 0.8 and 0.1 respectively ($d_c = 0.8, d_b = 0.1$), here the density of edges is the ratio of edges to all possible edges. The subspace of attributes of the three synthetic data sets are also shown in Table 6.1. The number of dimensionality of the subspace of each cluster is denoted as n_d and 2 dimensions are overlapping ($n_{od} = 2$). Each cluster contains nearly the same categories in its subspace attributes and only $r = 0.05$ of them are randomly generated with different other categories. The categories which do not belong to the subspace are randomly generated.

Table 6.1: Parameter Settings for Generating Synthetic Data Set

Datasets	Syn1	Syn2	Syn3
Sketch			
Structure Setting	$n_c = 200; n_o = 10;$ $d_c = 0.8; d_b = 0.1$	$n_c = 200; n_o = 10;$ $d_c = 0.8; d_b = 0.1$	$n_c = 200; n_o = 10;$ $d_c = 0.8; d_b = 0.1$
Attribute Setting	$n_d = 5; n_{od} = 2; r = 0.05$	$n_d = 8; n_{od} = 2; r = 0.05$	$n_d = 10; n_{od} = 2; r = 0.05$
Subspace Setting	$C1 : \{1, 2, \dots, 5\}$ $C2 : \{3, 4, \dots, 8\}$	$C1 : \{1, 2, \dots, 8\}$ $C2 : \{7, 8, \dots, 14\}$ $C3 : \{13, 14, \dots, 20\}$	$C1 : \{1, 2, \dots, 10\}$ $C2 : \{9, 10, \dots, 18\}$ $C3 : \{17, 18, \dots, 26\}$ $C4 : \{25, 26, \dots, 34\}$ $C5 : \{33, 34, \dots, 42\}$

Evaluation of Clustering. In this section, we evaluate IROC on three synthetic data sets and compare it to three relevant algorithms. PICS [3] is a compression based algorithm that clusters both vertices and binary attributes. BAGC [119] is a probability model based attributed graph partition method. DB-CSC [42] is a density based algorithm which aims at detecting dense clusters with a coherent subspace of attributes. It is designed for graphs with numerical attributes,

which chooses the attribute neighborhood if the distance of the attributes of two vertices is smaller than a threshold ϵ . The distance is defined as the maximum difference of all the attributes like $dist(x, y) = \max_{i \in \{1, \dots, T\}} |x[i] - y[i]|$. We transfer the categorical attributes of our synthetic data to integers for DB-CSC and set $\epsilon < 1$, which means the attribute neighbors with exactly the same category will be chosen. Due to the multiple clustering assignments, we adopt F-Measure to evaluate these algorithms on clustering vertices. F-Measure is computed as the harmonic mean of Precision and Recall. Precision measures the accuracy of the detected clusters and Recall measures whether all clusters are detected.

Table 6.2: Evaluation Overlapping Clusters of Synthetic Data Sets

Algorithms	Syn1			Syn2			Syn3		
	Pre.	Rec.	F-Measure	Pre.	Rec.	F-Measure	Pre.	Rec.	F-Measure
IROC	1	1	1	1	0.973	0.986	1	0.963	0.981
PICS	1	0.322	0.487	1	0.732	0.846	0.563	0.670	0.612
DB-CSC	0.895	0.826	0.859	—	—	—	—	—	—
BAGC	1	0.947	0.973	0.955	0.607	0.742	0.490	0.722	0.584

From Table 6.2, it is obvious that our proposed algorithm IROC outperforms the other algorithms. Generally, PICS and BAGC are not able to detect the overlapping parts of the graphs. DB-CSC outputs many small clusters with several overlapping vertices. However 6 parameters have to be set. Specifically, in *Syn1* IROC achieves perfect 2 clusters without any parameters, as shown in Figure 6.4(a). From Figure 6.4(b) the parameter-free algorithm PICS outputs 8 clusters, which splits two big cluster to several small ones. BAGC achieves 2 clusters, but it cannot detect their overlapping parts. DB-CSC outputs 19 clusters and some clusters contain less than ten vertices. We run DB-CSC with $\epsilon = 0.5$, $k_{min} = 4$, $min_{pts} = 5$, $r_{obj} = 0.1$, $r_{dim} = 0.1$ and $s_{min} = 1$. For the next synthetic data set *Syn2*, the overlap exists among all three clusters and dimensionality of each cluster is increased to 8. IROC achieves 3 perfect clusters. It also successfully detects the vertices which are assigned to all three clusters. PICS outputs 6 structural graph clusters without overlapping. BAGC produces 3 cluster, but again cannot find any overlap. DB-CSC achieves no result after adjusting 6 parameters several times and running the algorithm several days. The reason might be that DB-CSC is not applicable for dense graphs with higher

dimensional attributes. The last synthetic data set, *Syn3*, is more complex than the other two synthetic data sets before with 5 clusters and 4 of them overlapping. Similarly as before, IROC performs the best finding 5 good clusters and their respecting overlap. PICS partitions the vertices to 7 clusters. BAGC partitions the vertices to 5 clusters with no overlap. DB-CSC still cannot output any results after increasing the number of dimension of each cluster to 10.

Evaluation of Subspace. BAGC is not able to detect the subspaces of attributes. PICS is an algorithm which aims at clustering the binary attributes and not at finding the attributed subspaces of certain clusters. For example, Figure 6.4 shows the attributed matrix of *Syn1* after running IROC and PICS. Specifically, the size of the attributed matrix of IROC is 380×8 , where black points mean that the attribute is included in a subspace of a cluster. Two black blocks represent the clustering results of our IROC, which clearly shows the subspace of attributes and both overlaps on vertices and attributes. While the size of the attribute matrix of PICS is 380×35 . Black points mean that the vertices have corresponding categories and red lines demonstrate there are 5 attributes in the clusters. It is difficult to judge which attributed clusters are part of the subspace to represent the meaning of the cluster. In addition, subspaces of various clusters may contain overlap, while the clustering of the attributes is not able to achieve overlaps. Besides, it is hard to obtain the binary image of DB-CSC due to the fact that too many clusters with subspace are produced. Therefore, we compare IROC with DB-CSC by evaluating the F1-Measure of the subspaces. Obviously, IROC achieves perfect results on every synthetic data set. The F-Measure of DB-CSC on *Syn1* is 0.13, as there are many small clusters and the clustering results on the structural part does not lead to detect accurate subspaces. Besides, PICS obtains 10 attribute clusters on *Syn2* and 9 attribute clusters on *Syn3*. Similarly, they provide no information of which attribute clusters are part of the subspace to represent the meaning of the vertex clusters.

Efficiency. In the section, we test the runtime of IROC and its comparison algorithms on synthetic data set when varying both the number of vertices and the number of attributes, which are shown in Figure 6.5 and Figure 6.6 separately.

In Figure 6.5, the synthetic data sets contain two clusters with 10 overlapping vertices, and each subspace contains 5 attributes with 2 of them overlapped. We increase the number of nodes

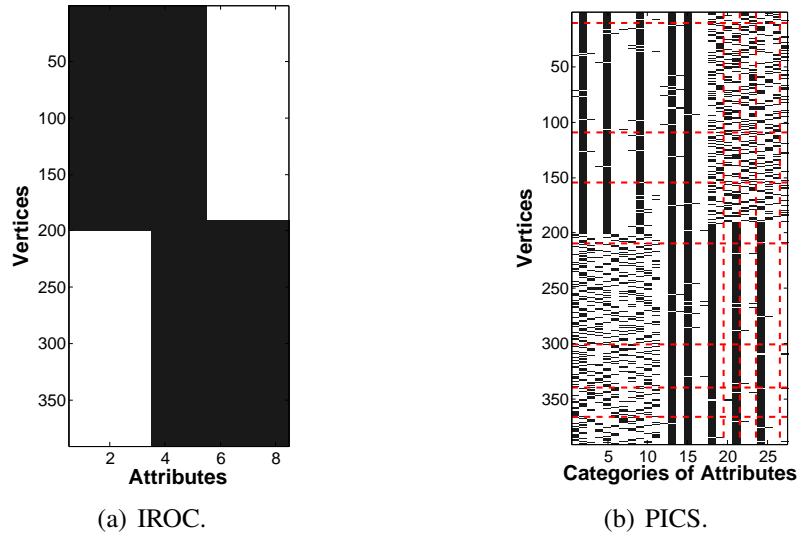


Figure 6.4: Clustering Results of Syn1 in Attributed Matrix.

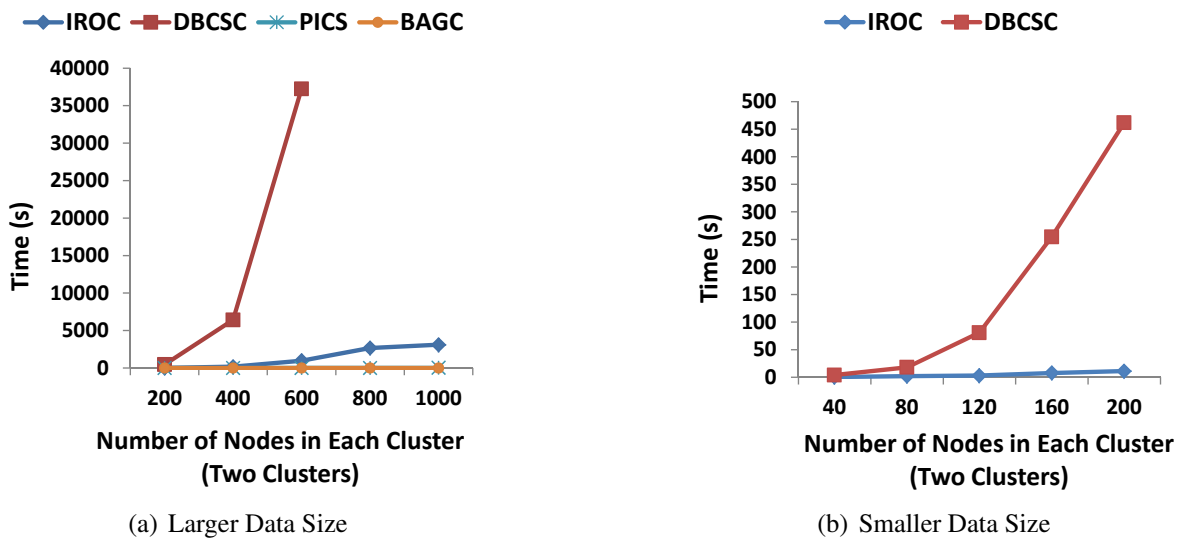


Figure 6.5: Run Time of Varying Vertices.

of each cluster from 200 to 1000. As shown in Figure 6.5 (a), IROC is slower than BAGC and PICS, but it is much faster and more stable than DB-CSC. In Figure 6.5 (a), DB-CSC do not show the complete results due to a too long runtime. Thus we compare it with IROC in the smaller data sets, as shown in Figure 6.5 (b), which also shows that IROC is faster and more stable than DB-CSC.

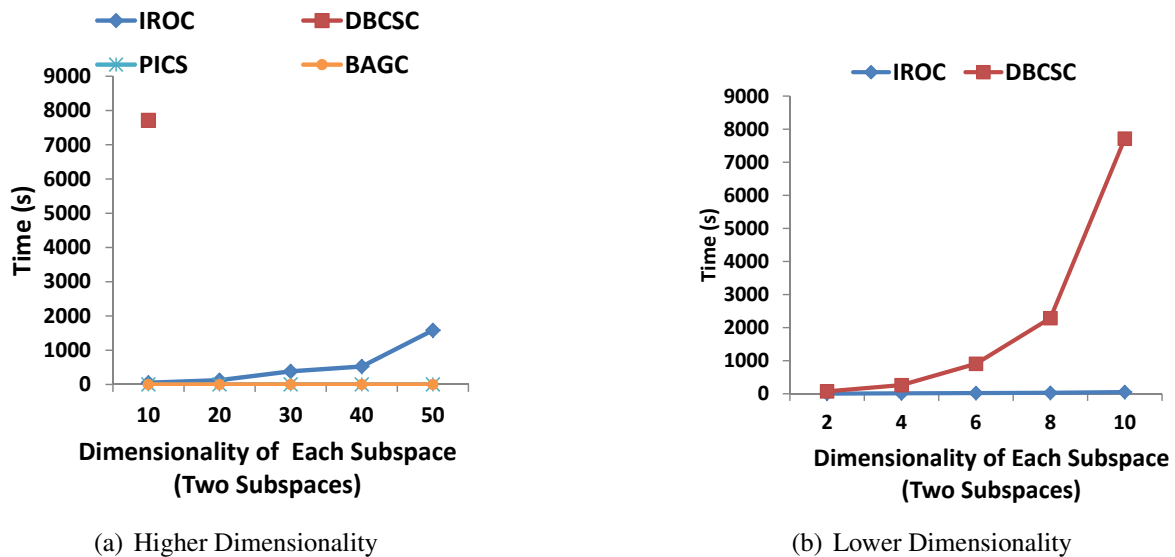


Figure 6.6: Run Time of Varying Dimensionality.

In Figure 6.6, the synthetic data sets contain two clusters with 10 overlapping vertices, and each cluster contains 200 nodes. We increase the number of attributes of each subspace from 10 to 50, and the two subspaces have 2 overlapping attributes. As shown in Figure 6.6 (a), IROC is slower than BAGC and PICS, but it is much faster and more stable than DB-CSC which is not able to show the complete results due to a too long run time. Similarly, we compare DB-CSC with IROC in a smaller dimensionality, as shown in Figure 6.6 (b), which also shows that IROC is faster and more stable than DB-CSC.

6.4.2 Real Data sets

Facebook Data. Facebook data sets are obtained from SNAP data sets [74]. Specifically, each Facebook data set is an ego-network of a selected node. For example, a network named “1912”

is generated by a node with id 1912 and all its neighbors. Characters of vertices include birthday, education information, language, hometown, work information .etc. Each kind of property contains several anonymity items. The original node attributes are in binary form, and every single item is considered as an attribute. The value “1” stands for the node that contains this item information and value “0” vice versa. In this section, we consider each type of property as attribute and the items in each property are defined as categories. In this section, we test two data sets: Ego-network “1912” and Ego-network “107”. However, for the same reason with *Syn3*, we are not able to get the results from DB-CSC after running the experiments for several days. Each data set is separated into several “circles”, but not all the vertices are included. That means only parts of the vertices’ label are available, thus we still use F-Measure to compare the algorithms.

There are 755 vertices in the ego-network “1912”, and each vertex has 22 attributes. Each attribute contains several categories, for example, there are 22 categories in attribute “birthday” and 19 categories in “education classes”. The data set has already been labeled as 46 circles. Obviously, Table 6.3 demonstrates that our proposed method achieves the best F-Measure value by generating 12 clusters. PICS generates 8 clusters. The result of BAGC is acquired by setting the number of clusters to 46, which is equals to the given number of clusters. Ego-network “107” contains 1045 vertices, 23 attributes and 9 circles. Table 6.3 demonstrates that our proposed method IROC achieves the best F-Measure value by generating 17 clusters. PICS partitions the vertices into 15 clusters, and the result of BAGC is acquired by setting the number of clusters as 9. Therefore, Table 6.3 demonstrates that IROC has a big advantage in detecting vertex clusters.

Table 6.3: F-Measure of Facebook Data Sets

Data sets	IROC	PICS	BAGC
<i>Ego – network</i> ”1912”	0.231	0.224	0.229
<i>Ego – network</i> ”107”	0.141	0.101	0.118

In addition, IROC discovers overlapping vertices between pairs of clusters. Take the Ego-network “1912” data as an example, the analysis of overlapping is shown in Table 6.4. Specifically, Figure 6.7 shows a diagram of the overlapping part between cluster C_2 and cluster C_{11} . There are four overlapping vertices “1941”, “2347”, “2468” and “2543” forming a clique. Other

vertices in C_2 and C_{11} are both closely connected to the clique. Therefore, the two clusters C_2 and C_{11} share the information of the overlapping part and it is reasonable to assign overlapping vertices to both clusters.

Table 6.4: Overlapping of Ego-network “1912”

Cluster ID	Overlapping Clusters ID(No.Vertices)
C_1	$C_5(1)$
C_2	$C_3(1), C_4(4), C_5(23), C_6(11), C_7(1), C_8(3), C_9(2), C_{10}(6), C_{11}(4), C_{12}(9)$
C_3	$C_6(1), C_{11}(15), C_{12}(4)$
C_4	$C_5(1), C_6(1), C_{11}(11), C_{12}(12)$
C_5	$C_6(1), C_7(1), C_9(1), C_{12}(3)$
C_6	$C_{10}(1), C_{12}(3)$
C_{10}	$C_{11}(1)$

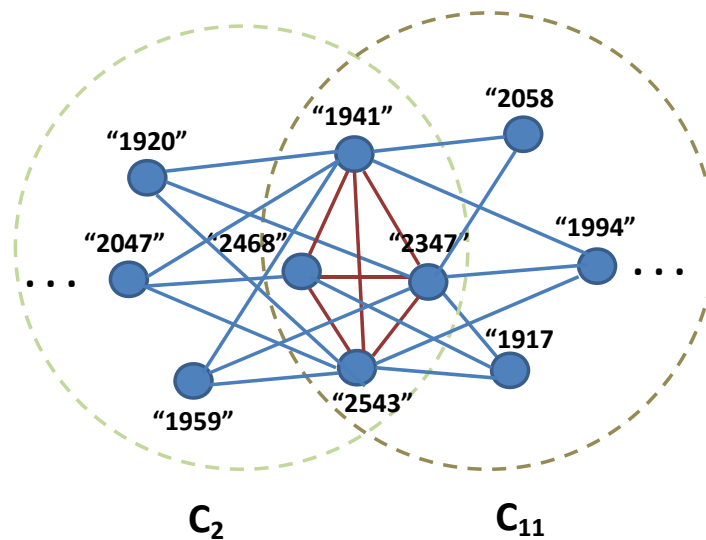


Figure 6.7: Overlapping Between Cluster 2 and Cluster 11 of Ego-network “1912”.

Betweenness centrality of a vertex is the ratio of the number of the shortest paths from v_m to v_n through vertex v to the number of all shortest paths from v_m to v_n which is shown in Eq. (2.3). Where betweenness centrality measures the ability of a human communicating with other humans in a social network [34], the vertex with high betweenness plays the role of bridge in graphs. We calculate betweenness centrality of all the vertices, and rank them in descending

order. Take the overlapping part of C_2 and C_{11} as an example, the rank of the betweenness values of “1941”, “2347”, “2468” and “2543” is 5, 4, 20 and 3 separately, which possesses really high betweenness centrality values. Most likely these four persons are very outgoing and sociable which is shown by this value. They communicate with various communities, so that they are assigned to multiple clusters. Besides, the betweenness values of all overlapping vertices are large, which implies that these vertices possess higher ability to communicate with other vertices thus should be assigned to multiple clusters.

Regarding the attributes, BAGC is not able to detect any subspace of clusters. PICS detects 8 attributes clusters in Ego-network “1912” and 11 attributes clusters in Ego-network “107”. Our IROC detects a subspace of attributes for each cluster. Table 6.5 shows the subspaces of the attributes of Ego-network “1912”. The table shows that overlaps also exist between these subspaces. For example, C_5 and C_6 have one vertex overlapping, and meanwhile their subspaces also overlap on attribute ”middle name”.

Table 6.5: Subspace detected by IROC of Ego-network “1912”

Cluster ID	Subspace
C_1	all 22 attributes
C_2	“middle name”
C_3	“middle name”, “work projects”
C_4	all 22 attributes
C_5	“middle name”, “work projects”
C_6	“middle name”
C_7	all 22 attributes
C_8	all 22 attributes
C_9	all 22 attributes
C_{10}	“work projects”
C_{11}	“work projects”
C_{12}	“middle name”

Google+ Data. Google+ data sets are obtained from the SNAP data sets [74] as well. Specifically, each Google+ data set is also an ego-network of a selected node. Similarly, we transfer the given binary features to categorical features which contain 6 attributes: gender, institution, job title, last name, place and university. Similarly, part of the labels are given as circles, thus we compare the F-Measure of the algorithms: IROC, PICS, BAGC and DBCSC.

Table 6.6: F-Measure of Google+ Data Sets

Data sets	IROC	PICS	BAGC	DBCSC
gp7	0.248	0.205	0.236	0.106
gp53	0.289	0.125	0.130	0.074

The data set “gp7” contains 1206 vertices and 6 attributes and the data set “gp53” contains 1084 vertices and 6 attributes. Table 6.6 shows that IROC achieves the best clustering result among all the algorithms on these two data sets. Specifically, take the data set “gp53” as an example, our proposed algorithm IROC detects 17 clusters with overlapping and without redundancy. Each cluster is provided with an attribute subspace which represents the meaning of the clusters. For example, people in one cluster are connected densely and all of them are from the same university, or people in one cluster are densely connected and all of them are from the same institution. Moreover, PICS outputs 12 vertices clusters and 6 attributes clusters. We set the number of cluster of BAGC to 4, which is equals to the given number of circles. DBCSC outputs 15 clusters parametrized with $\epsilon = 0.5$, $k_{min} = 2$, $min_{pts} = 3$, $r_{obj} = 0.3$, $r_{dim} = 0.3$, $s_{min} = 3$. All the 15 detected clusters are provided with a subspace. However, there are three clusters with exactly same vertices which contain redundant information.

6.5 Related Work and Discussion

Our proposed algorithm IROC is designed for clustering the attributed graph under the permission of overlapping in both vertices and the subspace of attributes. The related work therefore comprises two parts: attributed graph clustering and overlapping community detection methods.

6.5.1 Attributed Graph Clustering

Attributed graph is an extension of general graph by involving the attributes of the vertices. The key point of mining such complex data is to combine structural connections and characteristics of the vertices. Paper [127] augments graphs by considering each attribute as vertex. A Random walk is utilized on the augmented graph so as to create a unified similarity measure which com-

bines structural and attribute information. However, the algorithm is carried out in a K-Medoids framework and partitions the attributed graph. It also needs to input the number of clusters and parameters for the random walk, such as number of steps and the probability to go back. Moreover, paper [128] proposes an incremental algorithm to improve efficiency of [127]. Paper [119] proposes an algorithm named BAGC based on Bayesian probabilistic model to partition attributed graphs. Structural and attributed information are fused by the probabilistic model and clustering problem is transferred to a probabilistic inference problem. Similarly, the algorithm needs to input the number of clusters, and many other parameters to construct the necessary probability distribution. Obviously, these partition based methods can not detect any overlapping of clusters. And they do not find coherent attributed subspace of cluster. For these partitioning approaches, we choose BAGC as a comparison method.

Guennemann et al. proposes an algorithm named DB-CSC (Density-Based Combined Subspace Clustering) [42] which is based on the classical density-based clustering algorithm DBSCAN [29]. The new proposed DB-CSC inherits the advantages of DBSCAN which can detect clusters of arbitrary shapes and sizes. It defines a combined local neighborhood by finding vertices, which belong to the intersects of k -neighborhood of vertices and ϵ -neighborhood of subspace of the attributes. Based on the new defined neighborhood, some density related properties like high local density, local connected and maximality is defined, thus the fulfilled clusters can be detected. Instead of giving the number of clusters, DB-CSC needs parameters like ϵ -neighborhood, k -neighborhood and a minimum number $Minpts$. In order to remove redundant clusters of the above two algorithms, the authors proposed a definition which is used to calculate redundancy between clusters. After adopting the strategy of removing redundancy clusters, the combined new algorithms need more parameters, r_{obj} and r_{dim} which measure how much overlap between clusters is allowed without redundancy. Therefore, DB-CSC needs an isolate process and set several parameters to remove redundancy. If the parameters is set unproperly, redundancy still exists. In comparison, IROC obtains the non-redundant results without setting any parameters. Moreover, as mentioned in experiments part, judging from the distance defined in the chapter, DB-CSC is defined to deal with the attributed graphs with numerical attributes. But categorical attributes like gender, hobby .etc are common in social network data

set. Therefore, IROC is able to detect overlapping clusters of the categorical attributed graph and meanwhile find the coherent attribute subspace of each cluster.

PICS [3] is also a parameter free algorithm based on MDL principle. It is able to mine cohesive clusters from an attributed graph with similar connectivity patterns and homogeneous attributes. However, it can not detect any overlapping and it cluster the vertices and attributes separately. Thus it is hard to find out which subspace belongs to which clusters. Additionally, Sun et al. [105] proposes a model-based method to clustering heterogeneous information networks which are containing incomplete attributes and multiple link relations. Also marginally related to our method are the approaches [100, 79, 111] achieving numerous small cohesive subgraphs, which aim to discover a correlation between node attributes and small subgraphs. Paper [110] summarizes multi-relational attributed graphs by aggregating nodes by using selected attributes and relations.

6.5.2 Detecting Overlapping Communities

The key point of acquiring overlapping clusters is how to assign a vertex to multiple labels. In first instance, [86] reveals overlapping phenomena of complex networks in nature, and achieves overlapping communities by seeking k -cliques which contains overlapping vertices. Paper [125] proposes an algorithm based on matrix factorization which gives the soft clustering results. The assignment of each vertices is stored as probability in a matrix with a number of dimensions equal to the number of the community. By fuzzy allocation, overlap between communities is achieved. CONGA[37] proposed by Gregory is an algorithm which aims to detect overlapping communities by iteratively calculating two betweenness centrality based concepts “edge betweenness” and “split betweenness” of all edges and vertices respectively and removing the edge or splitting the vertex with the highest value until no edges remain. As betweenness centrality is a global measure of vertices in a graph, the calculation of the two concepts depends on counting the number of shortest paths of all pairs of vertices, which is really time consuming. In order to speed up the algorithm CONGA, the author proposes an algorithm named CONGO [38] by calculating local betweenness instead of global betweenness. In the new algorithm, a parameter h is added,

which is a threshold that the shortest path which is more than h is ignored. Thus the concepts only need to recalculate locally to save time complexity. Both CONGA and CONGO need user to predetermined the number of clusters k .

6.6 Chapter Conclusion

Summarizing this work, we introduced the first solution that is explicitly able to find meaningful overlapping in the graph structure as well as in its respective attribute subspace. We outlined the importance of these overlapping communities especially for attributed graphs and the information gain that can be received by it. Our method IROC applied the concept of information theoretic measures like entropy and an Minimum Description length (MDL) formula designed for this challenge to elegantly avoid a) redundancy in the attribute space as well as in the network itself and b) the need to set in an unsupervised setting typically unknown input parameters to achieve good results. Our experiments clearly showed that IROC is able to outperform all relevant comparison methods on synthetic data and on real world data of social networks.

Chapter 7

Conclusions and Future Research

In this thesis, we focus on adopting information theory to mine knowledge from various graph data: simple graph, bipartite graph and attributed graph. Four novel algorithms are proposed to fulfill different graph mining tasks, such as detecting interesting nodes, distinguishing graph structures, finding graph clusters, detecting hidden structures, predicting missing links. As some of the graph mining tasks are interrelated with each other, our proposed algorithms aim to achieve multiple tasks simultaneously. Moreover, Minimum Description Length (MDL) principle is the core technique of proposed algorithms, which leads the algorithms achieving parameter-free. The algorithms are implemented on both synthetic and real data sets and outperform the comparison methods.

Chapter 1 introduces many types of graph. In this thesis, we only discuss some of them. In the future, adopting information theory we are able to mine other types of graph, such as the weighted graph, the multi-relational graph or even more complex graph. And we can also apply the proposed algorithms to many real life field. For example, the brain network can be modeled as a simple graph, the relation between genes and proteins can be modeled as a bipartite graph. In the following, we summarize each proposed algorithm and point out the future work respectively.

7.1 Spotting Information-rich Nodes in Graphs

In real life, extracting interesting nodes from large graph is a significant task. And the task can be easily achieved by obtaining the background knowledge. However, the background knowledge is not always available. Info-spot is an algorithm which is proposed to detect the most interesting nodes in a large graph from the perspective of information theory. First of all, Info-spot gives a definition of the interesting node based on compressing link pattern of each node. An interesting node is defined as the node with a special or unique link pattern which can not be compressed. Furthermore, Info-spot is achieved by iteratively greedy merging pairs of nodes having the most similar link patterns. And the novel information theoretic based similarity is proposed to measure the difference between the link patterns of nodes in bits. Lastly, Info-spot is able to detect the interesting nodes without requiring any input parameters, assumptions or thresholds. A preliminary version of this work has been published in [47].

Info-spot is a greedy merging algorithm which is controlled by the Minimum Description Length principle. In ongoing and future work, one possible way to extend Info-spot is to adopt the information-theoretic framework for graph summarization which aims to detect summary graph and simplifies the larger graph. The other possible direction is to guide the nodes to merge as clusters and the graph clustering can be achieved.

7.2 Compression-based Graph Mining Exploiting Structure Primitives

Graph is a structural data, it is intuitive to mining graph from the perspective of structure. CXprime is proposed for mining graphs based on compressing three-node primitives which are the smallest substructures containing both star structure and triangular structure. Therefore, two types of coding schemes aiming to compress the graphs with frequent star structure and frequent triangular structure are generated respectively. Based on these two coding schemes of CXprime, the structure type of a graph as star-like or clique-like are able to be detected. Furthermore, guided by Minimum Description Length principle, CXprime is able to partition the graph into

clusters with star or triangular structures without setting the number of clusters. Additionally, unsupervised link prediction score of CXprime is generated based on the star structure and triangular structure, which is able to accurately predict missing link. To sum up, CXprime is a multi-functional algorithm and is able to achieve the tasks without setting input parameters. A preliminary version of this work has been published in [30].

In the future, one possible way to extend CXprime is to speed up the algorithm to manage very large network data. As CXprime need to count the number all three-node primitives which is time consuming. It is possible to adopt some fast counting algorithm as proposed in [95][61] to improve the efficiency. Besides, the other time consuming process is coding each entry of adjacency matrix, because it need to count the existing situations constantly. And parallel framework may be a possible way to solve the problem. Moreover, besides star structure and triangular structure, the other typical substructures such as tree-structure and line-structure also can be considered to included into the algorithm to detect more type of clusters.

7.3 Summarization-based Mining Bipartite Graphs

By modeling the two relational data as bipartite graph, we analyze such data from the perspective of structures. SCMiner is a compression based multi-functional algorithm for mining bipartite graph. Firstly, SCMiner transforms the large bipartite graph into a compact summary graph by merging the nodes with same link patterns. Secondly, adopting Minimum Description Length (MDL) principle to control the merging process, SCMiner is able to detect the truly relevant clusters of both node types simultaneously. Moreover, the detected compact summary graph also contains the essential relationships between both types of clusters thus revealing the hidden structure of the bipartite graph. Lastly, suspicious edges which are probably erroneous or missing can be also revealed in the algorithm. To sum up, all the tasks can be achieved by SCMiner simultaneously and automatically. A preliminary version of this work has been published in [32].

In ongoing and future work the intuitive way to extend SCMiner is to adopt the algorithm to mine knowlege from more complex graph like K-partite graphs and graphs with different edge

types. SCMiner focus on detecting clusters of each type of vertex which is the local clusters of the graph. It is more interesting to make SCMiner be able to meanwhile detect the globe communities of the graph. Moreover, we also consider to extend the basic idea of SCMine to the weighted graph so as to detect clusters, find hidden structures and the most important is to realize link prediction on weighted graph which not only predicts the missing edges but also the weights. However, it is a challenge to summarize the weighted graph as the weighted graph contains both structure information and weights information. It is hard to find the right distribution of the weights, thus the compression do not gain much. And lossy compression method may be a possible way to solve the problem.

7.4 Finding Overlapping Communities in Attributed Graphs

The key point of clustering an attributed graph is to combine and balance the structural information and attribute information. IROC is an attributed graph clustering algorithm based on information theory. Starting from generating many small subgraphs, IROC adopts a bottom up way to greedily merge the most similar small subgraphs. The novel information theoretic similarity is defined by considering both structure and attribute aspects. And guided by the Minimum Description Length (MDL) principle, IROC is able to find overlapping clusters as well as coherent attribute subspaces automatically. Meanwhile, comparing with many other overlapping clusters detection algorithms, IROC avoids the oversize overlapping and handle the redundancy. Parts of the material presented in this work have been submitted in [31].

IROC is confined to process the graph with the categorical attributes. In the ongoing future, one possible way to extend IROC is to make the algorithm be able to process all kinds of attributes like numerical or categorical. Comparing with the general graph, attributed graph is added with attribute information. We are also able to adopt information theory to mine knowledge from other complex graph, such as multi-relational graph, weighted graph or heterogeneous graph.

Bibliography

- [1] L. A. Adamic and E. Adar. Friends and neighbors on the web. *Social Networks*, 25(3):211–230, 2003.
- [2] L. Akoglu, M. McGlohon, and C. Faloutsos. oddball: Spotting anomalies in weighted graphs. In *PAKDD (2)*, pages 410–421, 2010.
- [3] L. Akoglu, H. Tong, B. Meeder, and C. Faloutsos. Pics: Parameter-free identification of cohesive subgroups in large attributed graphs. In *SDM*, pages 439–450, 2012.
- [4] R. Albert and A. lászló Barabási. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74(1):47–97, 2002.
- [5] A. Banerjee, I. S. Dhillon, J. Ghosh, S. Merugu, and D. S. Modha. A generalized maximum entropy approach to bregman co-clustering and matrix approximation. In *KDD*, pages 509–514, 2004.
- [6] S. T. Barnard and H. D. Simon. Fast multilevel implementation of recursive spectral bisection for partitioning unstructured problems. *Concurrency - Practice and Experience*, 6(2):101–117, 1994.
- [7] C. Böhm, C. Faloutsos, J.-Y. Pan, and C. Plant. Robust information-theoretic clustering. In *KDD*, pages 65–75, 2006.
- [8] C. Böhm, C. Faloutsos, and C. Plant. Outlier-robust clustering using independent components. In *SIGMOD*, pages 185–198, 2008.

- [9] C. Böhm, K. Haegler, N. S. Müller, and C. Plant. Coco: coding cost for parameter-free outlier detection. In *KDD*, pages 149–158, 2009.
- [10] P. Boldi and S. Vigna. The webgraph framework I: compression techniques. In *WWW*, pages 595–602, 2004.
- [11] I. Borg and P. J. Groenen. Modern multidimensional scaling: Theory and applications (2nd ed.). *New York: Springer-Verlag*, pages 207–212, 2005.
- [12] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. Lof: Identifying density-based local outliers. In *SIGMOD*, pages 93–104, 2000.
- [13] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks*, 30(1-7):107–117, 1998.
- [14] T. N. Bui and C. Jones. A heuristic for reducing fill-in in sparse matrix factorization. In *PPSC*, pages 445–452, 1993.
- [15] D. Chakrabarti. Autopart: Parameter-free graph partitioning and outlier detection. In *PKDD*, pages 112–124, 2004.
- [16] D. Chakrabarti and C. Faloutsos. Graph mining: Laws, generators, and algorithms. *ACM Comput. Surv.*, 38(1), 2006.
- [17] D. Chakrabarti, S. Papadimitriou, D. S. Modha, and C. Faloutsos. Fully automatic cross-associations. In *KDD*, pages 79–88, 2004.
- [18] C.-K. Cheng and Y.-C. A. Wei. An improved two-way partitioning algorithm with stable performance [vlsi]. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 10(12):1502–1511, 1991.
- [19] F. Chierichetti, R. Kumar, S. Lattanzi, M. Mitzenmacher, A. Panconesi, and P. Raghavan. On compressing social networks. In *KDD*, pages 219–228, 2009.

- [20] H. Cho, I. S. Dhillon, Y. Guan, and S. Sra. Minimum sum-squared residue co-clustering of gene expression data. In *SDM*, pages 114–125, 2004.
- [21] M. Coscia, G. Rossetti, F. Giannotti, and D. Pedreschi. Demon: a local-first discovery method for overlapping communities. In *KDD*, pages 615–623, 2012.
- [22] S. de Lin and H. Chalupsky. Unsupervised link discovery in multi-relational data via rarity analysis. In *ICDM*, pages 171–178, 2003.
- [23] I. S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *KDD*, pages 269–274, 2001.
- [24] I. S. Dhillon, Y. Guan, and B. Kulis. A unified view of kernel k-means, spectral clustering and graph cuts. Technical Report TR-04-25, University of Texas Dept. of Computer Science, 2005.
- [25] I. S. Dhillon, S. Mallela, and D. S. Modha. Information-theoretic co-clustering. In *KDD*, pages 89–98, 2003.
- [26] C. H. Q. Ding, X. He, H. Zhab, M. Gu, and H. D. Simon. A Min-max Cut Algorithm for Graph Partitioning and Data Clustering. In *ICDM*, pages 107–114, 2001.
- [27] W. Eberle and L. B. Holder. Discovering structural anomalies in graph-based data. In *ICDM Workshops*, pages 393–398, 2007.
- [28] P. Erdős and A. Rényi. On the evolution of random graphs. In *Publication of the mathematical institute of the hungarian academy of sciences*, pages 17–61, 1960.
- [29] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD*, pages 226–231, 1996.
- [30] J. Feng, X. He, N. Hubig, C. Böhm, and C. Plant. Compression-based graph mining exploiting structure primitives. In *ICDM*, pages 181–190, 2013.

- [31] J. Feng, X. He, N. Hubig, and C. Plant. Detection of overlapping communities in attributed graphs. In *Submitted for Publication*.
- [32] J. Feng, X. He, B. Konte, C. Böhm, and C. Plant. Summarization-based mining bipartite graphs. In *KDD*, pages 1249–1257, 2012.
- [33] C. M. Fiduccia and R. M. Mattheyses. A linear-time heuristic for improving network partitions. In *DAC*, pages 175–181, 1982.
- [34] L. C. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, 40(1):35–41, 1977.
- [35] A. George and J. W. Liu. *Computer Solution of Large Sparse Positive Definite*. Prentice Hall Professional Technical Reference, 1981.
- [36] T. George and S. Merugu. A scalable collaborative filtering framework based on co-clustering. In *ICDM*, pages 625–628, 2005.
- [37] S. Gregory. An algorithm to find overlapping community structure in networks. In *PKDD*, pages 91–102, 2007.
- [38] S. Gregory. A fast algorithm to find overlapping communities in networks. In *ECML/PKDD (1)*, pages 408–423, 2008.
- [39] S. Gregory. Finding overlapping communities in networks by label propagation. *New Journal of Physics*, 12(10):103018, 2010.
- [40] S. Gregory. Fuzzy overlapping communities in networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2011, P02017.
- [41] P. Grünwald. A tutorial introduction to the minimum description length principle. In *Advances in Minimum Description Length: Theory and Applications*. 2005. MIT Press, 2005.

- [42] S. Günnemann, B. Boden, and T. Seidl. Db-csc: A density-based approach for subspace clustering in graphs with feature vectors. In *ECML/PKDD (1)*, pages 565–580, 2011.
- [43] N. Guttman-Beck and R. Hassin. Approximation algorithms for minimum k-cut. *Algorithmica*, 27(2):198–207, 2000.
- [44] J. Hao and J. B. Orlin. A faster algorithm for finding the minimum cut in a graph. In *SODA*, pages 165–174, 1992.
- [45] M. A. Hasan, V. Chaoji, S. Salem, and M. Zaki. Link prediction using supervised learning. In *SDM Workshop on Link Analysis, Counterterrorism and Security*, 2006.
- [46] D. Hawkins. *Identification of Outliers*. Chapman and Hall, London, 1980.
- [47] X. He, J. Feng, and C. Plant. Automatically spotting information-rich nodes in graphs. In *ICDM Workshops*, pages 941–948, 2011.
- [48] K. Henderson, B. Gallagher, T. Eliassi-Rad, H. Tong, S. Basu, L. Akoglu, D. Koutra, C. Faloutsos, and L. Li. Rolx: structural role extraction and mining in large graphs. In *KDD*, pages 1231–1239, 2012.
- [49] B. Hendrickson and R. Leland. A multilevel algorithm for partitioning graphs. In *ACM/IEEE Conference on Supercomputing*, 1995.
- [50] L. B. Holder, D. J. Cook, and S. Djoko. Substructure discovery in the subdue system. In *KDD Workshop on AAI*, pages 169–180, 1994.
- [51] S. Ihara. *Information theory for continuous systems*. World Scientific, 1993.
- [52] A. Inokuchi, T. Washio, and H. Motoda. An apriori-based algorithm for mining frequent substructures from graph data. In *PKDD*, pages 13–23, 2000.
- [53] G. Jeh and J. Widom. Simrank: a measure of structural-context similarity. In *KDD*, pages 538–543, 2002.

- [54] U. Kang and C. Faloutsos. Beyond 'caveman communities': Hubs and spokes for graph compression and mining. In *ICDM*, pages 300–309, 2011.
- [55] G. Karypis and V. Kumar. Multilevel graph partitioning schemes. In *ICPP (3)*, pages 113–122, 1995.
- [56] G. Karypis and V. Kumar. Multilevel k -way hypergraph partitioning. In *DAC*, pages 343–348, 1999.
- [57] H. Kashima and N. Abe. A parameterized probabilistic model of network evolution for supervised link prediction. In *ICDM*, pages 340–349, 2006.
- [58] L. Katz. A new status index derived from sociometric analysis. *PSYCHOMETRIKA*, 18(1):39–43, 1953.
- [59] E. M. Knorr and R. T. Ng. Algorithms for mining distance-based outliers in large datasets. In *VLDB*, pages 392–403, 1998.
- [60] D. E. Knuth. *The Stanford GraphBase: A Platform for Combinatorial Computing*. Addison-Wesley, Reading, MA, 1993.
- [61] M. N. Kolountzakis, G. L. Miller, R. Peng, and C. E. Tsourakakis. Efficient triangle counting in large graphs via degree-based vertex partitioning. *Internet Mathematics*, 8(1):161–185, 2011.
- [62] J. Kunegis and A. Lommatzsch. Learning spectral graph transformations for link prediction. In *ICML*, page 71, 2009.
- [63] J. Kunegis, E. W. D. Luca, and S. Albayrak. The link prediction problem in bipartite networks. In *IPMU*, pages 380–389, 2010.
- [64] M. Kuramochi and G. Karypis. Frequent subgraph discovery. In *ICDM*, pages 313–320, 2001.

- [65] M. D. Lee. Determining the dimensionality of multidimensional scaling representations for cognitive modeling. *Journal of Mathematical Psychology*, 45:149–166, 2001.
- [66] D. Liben-Nowell and J. M. Kleinberg. The link prediction problem for social networks. In *CIKM*, pages 556–559, 2003.
- [67] R. Lichtenwalter, J. T. Lussier, and N. V. Chawla. New perspectives and methods in link prediction. In *KDD*, pages 243–252, 2010.
- [68] Y.-R. Lin, J. Sun, P. Castro, R. Konuru, H. Sundaram, and A. Kelliher. Metafac: Community discovery via relational hypergraph factorization. In *KDD*, pages 527–536, 2009.
- [69] J. Liu and T. Liu. Detecting community structure in complex networks using simulated annealing with k-means algorithms. *Physica A: Statistical Mechanics and its Applications*, 389(11):2300–2309, 2010.
- [70] B. Long, X. Wu, Z. M. Zhang, P. S. Yu, and P. S. Yu. Unsupervised learning on k-partite graphs. In *KDD*, pages 317–326, 2006.
- [71] B. Long, Z. M. Zhang, P. S. Yu, and P. S. Yu. Co-clustering by block value decomposition. In *KDD*, pages 635–640, 2005.
- [72] L. Lü and T. Zhou. Link prediction in complex networks: A survey. *Physica A*, 390(6):11501170, 2011.
- [73] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In *In Proceedings of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297, 1967.
- [74] J. J. McAuley and J. Leskovec. Learning to discover social circles in ego networks. In *NIPS*, pages 548–556, 2012.
- [75] M.E.J.Newman. Clustering and preferential attachment in growing networks. *Physical Review E*, 64(2):025102, 2001.

- [76] G. L. Miller, S.-H. Teng, and S. A. Vavasis. A unified geometric approach to graph separators. In *FOCS*, pages 538–547, 1991.
- [77] M. Mitzenmacher. A brief history of generative models for power law and lognormal distributions. *Internet Maths*, 1(2):226–251, 2003.
- [78] H. D. K. Moonesinghe and P.-N. Tan. Outrank: a graph-based outlier detection framework using random walk. *International Journal on Artificial Intelligence Tools*, 17(1):19–36, 2008.
- [79] F. Moser, R. Colak, A. Rafiey, and M. Ester. Mining cohesive patterns from graphs with feature vectors. In *SDM*, pages 593–604, 2009.
- [80] N. Mueller, K. Haegler, J. Shao, C. Plant, and C. Böhm. Weighted graph compression for parameter-free clustering with pacco. In *SDM*, pages 932–943, 2011.
- [81] S. Navlakha, R. Rastogi, and N. Shrivastava. Graph summarization with bounded error. In *SIGMOD*, pages 419–432, 2008.
- [82] M. E. Newman. Modularity and community structure in networks. *Proc Natl Acad Sci U S A*, 103(23):8577–8582, 2006.
- [83] M. E. J. Newman. Finding community structure in networks using the eigenvectors of matrices. *Physical review E*, 74(3):036104, 2006.
- [84] C. C. Noble and D. J. Cook. Graph-based anomaly detection. In *KDD*, pages 631–636, 2003.
- [85] G. K. Orman, V. Labatut, and H. Cherifi. An empirical study of the relation between community structure and transitivity. *Studies in Computational Intelligence*, 424:99–110, 2013.
- [86] G. Palla, I. Dernyi, I. Farkas, and T. Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435(7043):814–818, 2005.

- [87] S. Papadimitriou, H. Kitagawa, P. B. Gibbons, and C. Faloutsos. Loci: Fast outlier detection using the local correlation integral. In *ICDE*, pages 315–326, 2003.
- [88] 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, 1990.
- [89] D. M. W. Powers. Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness & Correlation. Technical Report SIE-07-001, School of Informatics and Engineering, Flinders University, 2007.
- [90] J. Rissanen. Modeling by shortest data description. *Automatica*, 14(5):465–471, 1978.
- [91] J. Rissanen. An introduction to the mdl principle. Technical report, Helsinki Institute for Information Technology, 2005.
- [92] J. Rissanen. *Information and Complexity in Statistical Modeling*. Springer, 2007.
- [93] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., 1986.
- [94] T. Schank and D. Wagner. Approximating clustering coefficient and transitivity. *Journal of Graph Algorithms and Applications*, 9(2):265–275, 2005.
- [95] T. Schank and D. Wagner. Finding, counting and listing all triangles in large graphs, an experimental study. In *WEA*, pages 606–609, 2005.
- [96] H. Shan and A. Banerjee. Bayesian co-clustering. In *ICDM*, pages 530–539, 2008.
- [97] C. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27(3):379–423, 1948.
- [98] N. Shervashidze, S. V. N. Vishwanathan, T. Petri, K. Mehlhorn, and K. M. Borgwardt. Efficient graphlet kernels for large graph comparison. In *AISTATS*, pages 488–495, 2009.
- [99] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 1997.

-
- [100] A. Silva, W. M. Jr., and M. J. Zaki. Mining attribute-structure correlated patterns in large attributed graphs. *PVLDB*, 5(5):466–477, 2012.
- [101] A. Stolcke and S. M. Omohundro. Hidden markov model induction by bayesian model merging. In *NIPS*, pages 11–18, 1992.
- [102] A. Stolcke and S. M. Omohundro. Inducing probabilistic grammars by bayesian model merging. In *ICGI*, pages 106–118, 1994.
- [103] A. Strehl, J. Ghosh, and C. Cardie. Cluster ensembles - a knowledge reuse framework for combining multiple partitions. *Journal of Machine Learning Research*, 3:583–617, 2002.
- [104] J. Sun, H. Qu, D. Chakrabarti, and C. Faloutsos. Neighborhood formation and anomaly detection in bipartite graphs. In *ICDM*, pages 418–425, 2005.
- [105] Y. Sun, C. C. Aggarwal, and J. Han. Relation strength-aware clustering of heterogeneous information networks with incomplete attributes. *PVLDB*, 5(5):394–405, 2012.
- [106] Y. Sun and J. Han. Mining heterogeneous information networks: a structural analysis approach. *SIGKDD Explorations*, 14(2):20–28, 2012.
- [107] Y. Sun, J. Han, P. Zhao, Z. Yin, H. Cheng, and T. Wu. Rankclus: integrating clustering with ranking for heterogeneous information network analysis. In *EDBT*, pages 565–576, 2009.
- [108] Y. Sun, Y. Yu, and J. Han. Ranking-based clustering of heterogeneous information networks with star network schema. In *KDD*, pages 797–806, 2009.
- [109] W. Tang, Z. Lu, and I. S. Dhillon. Clustering with multiple graphs. In *ICDM*, pages 1016–1021, 2009.
- [110] Y. Tian, R. A. Hankins, and J. M. Patel. Efficient aggregation for graph summarization. In *SIGMOD*, pages 567–580, 2008.

- [111] H. Tong, C. Faloutsos, B. Gallagher, and T. Eliassi-Rad. Fast best-effort pattern matching in large attributed graphs. In *KDD*, pages 737–746, 2007.
- [112] S. van Dongen. *Graph Clustering by Flow Simulation*. Dissertation, University of Utrecht, 2000.
- [113] M. van Leeuwen, J. Vreeken, and A. Siebes. Compression picks item sets that matter. In *PKDD*, pages 585–592, 2006.
- [114] N. X. Vinh, J. Epps, and J. Bailey. Information theoretic measures for clusterings comparison: is a correction for chance necessary? In *ICML*, pages 1073–1080, 2009.
- [115] C. Wang, V. Satuluri, and S. Parthasarathy. Local probabilistic models for link prediction. In *ICDM*, pages 322–331, 2007.
- [116] X. F. Wang and G. Chen. Complex networks: small-world, scale-free and beyond. *Circuits and Systems Magazine*, 3(1):6–20, 2003.
- [117] D. Watts and S. Strogatz. Collective dynamics of ‘small-world’ networks. *Nature*, 393(6684):440–442, 1998.
- [118] J. Xie, S. Kelley, and B. K. Szymanski. Overlapping community detection in networks: The state-of-the-art and comparative study. *ACM Computing Surveys*, 45(4):43:1–43:35, 2013.
- [119] Z. Xu, Y. Ke, Y. Wang, H. Cheng, and J. Cheng. A model-based approach to attributed graph clustering. In *SIGMOD*, pages 505–516, 2012.
- [120] X. Yan and J. Han. gspan: Graph-based substructure pattern mining. In *ICDM*, pages 721–724, 2002.
- [121] J. Yang and J. Leskovec. Overlapping community detection at scale: a nonnegative matrix factorization approach. In *WSDM*, pages 587–596, 2013.
- [122] R. W. Yeung. *A First Course in Information Theory*. Springer, 2002.

-
- [123] W. W. Zachary. An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, 33:452–473, 1977.
- [124] N. Zhang, Y. Tian, and J. M. Patel. Discovery-driven graph summarization. In *ICDE*, pages 880–891, 2010.
- [125] Y. Zhang and D.-Y. Yeung. Overlapping community detection via bounded nonnegative matrix tri-factorization. In *KDD*, pages 606–614, 2012.
- [126] Y. Zhao and G. Karypis. Criterion functions for document clustering: Experiments and analysis. Technical report, 2002.
- [127] Y. Zhou, H. Cheng, and J. X. Yu. Graph clustering based on structural/attribute similarities. *PVLDB*, 2(1):718–729, 2009.
- [128] Y. Zhou, H. Cheng, and J. X. Yu. Clustering large attributed graphs: An efficient incremental approach. In *ICDM*, pages 689–698, 2010.

Acknowledgments

Time flies, four years of PhD study in general, such as a flash gone. There are sweetness and bitterness during this period. Here, I would like to express my gratitude to all those who helped me during this special time in Germany.

My deepest gratitude goes first and foremost to Prof. Dr. Christian Böhm, my supervisor, for all the support and encouragement he gave me during my research. Without his guidance and constant feedback this PhD would not have been achievable. Second, I would like to express my heartfelt gratitude to Dr. Claudia Plant for her scientific advice and knowledge and many insightful discussions and suggestions. She remains my best role model for a scientist. I am also greatly thankful to Prof. Anthony K. H. Tung, who takes time out of his busy schedule to review my thesis.

I thank my current and past colleagues at the data mining group of LMU: Xiao He, Son Mai Thai, Bettina Konte, Sebastian Goebel, Dr. Junming Shao, Qinli Yang, Dr. Bianca Wackersreuther, Dr. Annahita Oswald, Wei Ye, Linfei Zhou, Can Altinigneli, Frank Fiedler, Peter Wackersreuther and Andrew Zherdinand. And I also thank the colleagues at iKDD group of Helmholtz Zentrum München: Nina Hubig, Sam Maurus, Annika Tonch, Dr. Wolfgang zu Castell, Dr. David Endesfelder. Working with all of these people is a great learning experience for me. I want to express my deep appreciation for the help and support they have offered me during my PhD study. Furthermore, I want to thank Susanne Grienberger, Sandra Mayer and Franz Krojer for their kindly supports and help.

I gratefully acknowledge the China Scholarship Council and The University of Munich (CSC-LMU joint Scholarship) for providing me the financial support for my four years PhD work. And thanks also go to the people who are working for the program.

It is an honor for me to join the LMU-Mentoring program. I want to thank Prof. Dr. Francesca Biagini who offers me a lot of support. I also give thanks to the people who work for this program.

I would like to thank my friends in the Munich for all the great times that we have shared. I am deeply thankful to my beloved family for their loving considerations and great confidence in me all through these years. Especially, thanks to my coming child who has been with me during the thesis writing.

Jing Feng

Munich, Germany

November, 2014

Eidesstattliche Versicherung

(Siehe Promotionsordnung vom 12.07.11, § 8, Abs. 2 Pkt. .5.)

Hiermit erkläre ich an Eidesstatt, dass die Dissertation von mir selbstständig, ohne unerlaubte Beihilfe angefertigt ist.

Feng, Jing

Name, Vorname

Munich, 11.06.2015

Ort, Datum

Feng, Jing

Unterschrift Doktorand/in

Formular 3.2