DISCOVERING PATTERNS IN COMPLEX NETWORKS WITH APPLICATIONS TO LINK ANALYSIS AND CLUSTERING

LUN HU

Ph.D

The Hong Kong Polytechnic University

2015

The Hong Kong Polytechnic University

Department of Computing

Discovering Patterns in Complex Networks with Applications to Link Analysis and Clustering

Lun Hu

A thesis submitted in partial fulfillment of

the requirements for the degree of

Doctor of Philosophy

October 2014

CERTIFICATE OF ORIGINALITY

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgement has been made in the text.

_____(Signed)

Lun Hu (Name of student)

Abstract

A network consists of a set of objects and their connections and a complex network is a network that has a non-trivial topology. A computational technique that can discover interesting patterns in complex networks can have many applications in a variety of research areas. For example, it can be used to discover protein complexes in protein-protein interaction networks, or to identify online user communities in social networks. Networks can be represented as graphs with vertices representing objects and edges representing connections between objects. Hence, to discover patterns in networks, graph mining techniques have therefore been used. For many of them to work effectively, patterns are required to have specific topological properties in terms of density, maximal *k*-cliques, or betweenness centrality. But the attributes associated with the objects in a complex network are usually ignored, or treated separately, during the graph mining process. According to empirical studies on complex networks, associations are believed to be existed between the attributes of objects and the links between objects and thus they may provide valuable information for discovering of interesting graph patterns. In this regard, we propose in this thesis a technique that can discover associative patterns from complex networks by taking into consideration the associations between attribute and topology information during the pattern discovery process. This technique works with what are called attributed graphs (AGs). Associated with each vertex in such a graph is an attribute set where each of attribute can take more than one value.

Obviously, to discover associative patterns is to discover regularities between attribute and topology information of AGs. A simple but feasible way to represent them is to make use of pairwise attribute values that are significantly observed in connecting vertices in the AG given. That is to say, if the frequency of co-occurrence of the respective attribute values in two connecting vertices is significantly higher, the co-occurrence of the two attribute values is the associative pattern of interest. Hence, for two attribute values, to determine if the frequency of their co-occurrences is significantly higher, we make use of statistical analysis to determine if the conditional probability of one attribute value given the other is significantly higher from the a

priori probability of the attribute value occurring irrespective of other attribute values. If the difference is verified to be statistically significant, then the frequency of co-occurrences of the two attribute values can be considered as significantly higher. In this case, the co-occurrence of these two values constitutes an associative pattern. Once such an associative pattern is identified, we further make use of an information theoretic measure to indicate how significant this pattern is. Hence, for two inter-connected objects that are represented as two vertices connected by a link in an AG, the association between them can be determined by the number and the amount of significances of association patterns found in between them. The proposed technique can hence discover associative patterns in AGs based on both topological and attribute information. Then a Degree of Association (DOA) measure is introduced to compute the association between vertices based on the amount and the significances of associative patterns found in their attributes. The introduction of associative patterns allows us to fully utilize the potential knowledge in AGs in an efficient way and we can use them to tackle problems in a diversity of graph mining problems. For performance evaluation, we have used it to solve problems in link analysis and graph clustering.

For link analysis, associative patterns have been used to predict Protein-Protein Interactions (PPIs) in PPINs based on the protein sequences as attributes for the proteins in the network. An algorithm, VLASPD, has been developed based on the proposed technique to consider variable-length segments of each pair of interacting protein sequences to determine the association relationship that exist between these proteins. Unlike other sequence-based approach, VLASPD is able to discover patterns in interacting proteins by considering association between variable-length segments. As a result, it is able to make use of such patterns to more accurately predict if two proteins may interact with each other. We have tested VLASPD with different real data sets and the experimental results show that VLASPD can predict PPIs accurately and can be a promising approach for PPI prediction.

For AG clustering, we first propose a fuzzy-based clustering approach, namely FC-AG, by combining the topology and attribute information of AGs with the DOA

measure. The adoption of fuzzy clustering allows FC-AG to identify clusters in a natural manner. However, since there are also applications whose number of clusters is unknown, we further develop an unsupervised clustering algorithm, namely MCL-AG, to identify clusters through a markov clustering process. Integrated with the DOA measure, MCL-AG is able to discover dense graph clusters consisting of vertices whose attribute values may have significantly closer association with each other. However, based on the experimental results of MCL-AG, we note that vertices in the same cluster have not to be similar over all attributes. Therefore, if we have a way to perform the unsupervised clustering by resting on attributes that are more similar while ignoring those with less similarity, clusters can be identified more accurately and efficiently. To do so, we propose an algorithm, namely CAP-AG, so that the attribute preferences can be considered during clustering. To evaluate the performance of FC-AG, MCL-AG and CAP-AG, we have applied them to several practical problems, including document classification, social community identification and the prediction of protein complexes. The experimental results show the promising performances of these three approaches.

List of Publications

Journal Papers

1. Hu, L., & Chan, Keith C. C. (2015). A fuzzy-based clustering approach for network data using content relevance and link structure. Submitted to *IEEE Transactions on Fuzzy Systems* (2nd Revision).
2. Hu, L., & Chan, Keith C. C. (2015). Prediction of Protein-protein Interactions Using Coevolutionary Features in Sequence Information. Submitted to *IEEE/ACM Transactions on Computational Biology and Bioinformatics* (2nd Revision).
3. Hu, L., & Chan, Keith C. C. (2015). A Density-based Clustering Approach for Identifying Overlapping Protein Complexes with Functional Preferences. Accepted by *BMC Bioinformatics*.
4. Hu, L., & Chan, Keith C. C. (2015). Discovering Variable-Length Patterns in Protein Sequences for Protein-protein Interaction Prediction. Accepted by *IEEE Transactions on NanoBioscience*.
5. Hu, A. L., & Chan, Keith C. C. (2013). Utilizing Both Topological and Attribute Information for Protein Complex Identification in PPI Networks. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, *10*(3), 780-792.

Conference Papers

1. Hu, A. L., & Chan, Keith C. C. (2014). Identifying protein complexes using attribute preferences. *Proceedings of 13th International Workshop on Data Mining in Bioinformatics (BioKDD'14)*, New York, USA.

Acknowledgements

I would like to express the deepest appreciation to my supervisor Professor Keith C. C. Chan, who has shown the attitude and the substance of a genius: he continually and persuasively conveyed a spirit of adventure in regard to research. Without his supervision and constant help this thesis would not have been completed.

I would like to thank my friends: Hua Luo, Wenlin Qu, Xiao Zhang, and Xinyao Ma. We have been being friends since I came to Hong Kong seven years ago and I am fortunate in witnessing their every important moment in life, like wedding and the birth of their children. In addition, many thanks to my colleagues: Dr. Zhuhong You, Dr. Xin Luo, Pengwei Hu, Zhonglei Gu, Weiming Luo, Yufei Wang, Zhizhao Feng and Tiantian He, for the stimulating discussions, and for all the fun we have had in the last four years.

A special thanks to my parents for all of the sacrifices that you've made on my behalf. Their love for me was what sustained me thus far. I would also like to thank my beloved wife Zisang Hu. We both know that we are striving towards to a happiness life of us. To them I dedicate this thesis.

Table of Contents

List of Figures

List of Tables

Chapter 1 Introduction

Networks are composed of interconnected objects and they are reckoned as an advanced model to describe the topological structure constructed from the connections of objects. When put in the context of graph, objects in a network are referred to as vertices while the connections among them are referred to as links, or edges. Recently, as both sources and techniques of retrieving information have been undergoing rapid development, the complexity of network data obtained is more complicated than ever in terms of the amount of information carried.

Among a variety of complex networks, attributed graphs (AGs) are of substantial interest because of its ability of allowing attributes to be associated with vertices. Such ability can enrich the content information we can find in networks, thus providing an alternative way for us to analyze network data. A considerable number of examples of AGs can be identified in the real world. For example, Protein-Protein Interaction Networks (PPINs) can be represented as attributed graphs if protein knowledge is provided for proteins as well. As another example, social networks are able to be modeled as attributes graphs if personal profiles are available.

Hence, for AGs, there are two kinds of information involved, one is the topological information of graph and the other is the attribute information of vertices in the graph. Both of these two kinds of information are believed to be significant for us to understand and analyze AGs and only considering either of them could probably lead to unsatisfactory performances for specific applications related to AGs, such as link analysis and graph clustering. However, regardless of the purposes of different applications, most of existing approaches proposed for these applications only make use of topological information to perform their tasks while failing to consider the attribute information. A major reason for the missing attribute information in these approaches is ascribed to the lack of AGs when they were under development. Hence,

the non-trivial attribute information of AGs raises new challenges, as we have to consider the attribute information in addition to the topological information when addressing applications where AGs are involved.

In this thesis, we will concentrate on tackle such challenge so that both topological and attribute information can be appropriately utilized for AG applications. Regarding the process of these two kinds of information in AGs, although some attempts [19], [84], [95], [97], [98], [103], [104] have been made, we note that they normally process these two kinds of information separately and then integrate them in a unified clustering framework. Specifically speaking, for the application of graph clustering, distance-based approaches [19], [103], [104] design different distance measures to compute the similarity of attribute information and that of topology information respectively while model-based approaches [84], [95], [97], [98] adopt different probabilistic models to estimate the likelihood of being associated with a certain attribute value and that of being connected by a certain vertex. In this regard, existing approaches proposed for AG applications cannot disclose the association between attribute and topology information, which has been verified by the empirical studies of social networks with attribute information [2], [86]. Therefore, it is intuitive to believe that both attribute and topology information can be utilized more efficiently if there is a way to discover the association behind them.

To disclose the association between attribute and topology information, we consider this problem from an alternative view. In particular, we intend to discover associative patterns composed of pairwise attribute values that are frequently observed in the adjacent vertices of AG with statistical significance, as these associative patterns are the regularities observed in AG by taking both attribute and topology information into consideration and we believe they are eligible to represent the association between these two kinds of information. Therefore, in this thesis, we propose an algorithm of discovering associative patterns in AG with some statistical knowledge. In addition to the discovery of associative patterns, the proposed algorithm can also measure the strengths of associative patterns from the perspective of information theory.

To demonstrate the advantage of associative patterns in AG, we apply them to the AG applications of link analysis and graph clustering. In particular, for link analysis, we have studied the problem of predicting links and then developed an approach, namely VLSPAD, to perform the prediction task by making use of associative patterns. To evaluate the efficiency of VLSPAD, we apply it to the application of predicting Protein-Protein Interactions (PPIs) from PPINs and the experimental results show the promising performance of VLSPAD.

Regarding graph clustering, we have been engaging in identifying clusters from AG by utilizing both of attribute and topology information simultaneously and the discovery of associative patterns provides us a solid theoretical basis to do so. First of all, a Degree of Association (DOA) measure is introduced so that the association between pairwise vertices can be quantified based on the amount and the significance of associative patterns found in their attributes. With this DOA measure, three different approaches, i.e., FC-AG, MCL-AG and CAP-AG, have been proposed to tackle the problem of AG clustering. In particular, FC-AG is a fuzzy-based clustering approach that utilizes both of the topology and attribute information to estimate the fuzziness of memberships for each of vertices. Unlike FC-AG that requires to determine the number of clusters in advance because of the adoption of fuzzy clustering, MCL-AG makes use of markov clustering to partition an AG into several dense subgraphs by considering the results of DOA measure and then identify clusters from these subgraphs. From the experiment results of MCL-AG, we note that it is not necessary for vertices in the same cluster to be similar in all attributes. Motivated by this observation, we also propose a clustering approach using attribute preferences, namely CAP-AG. To evaluate the performances of these three approaches, we have conducted extensive experiments with the comparison to the state-of-the-art clustering approaches and the experiment results show that the proposed approaches have a promising performance when identifying clusters from AGs.

The rest of this chapter is organized as follows. Section 1.1 introduces the motivations behind the introduction of associative patterns as well as the approaches proposed to

solve the applications of link analysis and clustering based on associative patterns. In Section 1.2, a brief description about the applications of link analysis and graph clustering for AGs is given. The contributions made by the works in this thesis are presented in Section 1.3. The last section of this chapter, i.e., Section 1.4, describe the organization of thesis so as to give an overall view of this thesis.

1.1 Motivations

The motivations of the works made in this thesis are mainly from two aspects: 1) *Theoretical Motivation* for pattern discovery in AG, and 2) *Approach Innovations* for the applications of link analysis and clustering. Hence, the rest of this section will be unfolded from these two aspects.

1.1.1 Theoretical Motivation

Regarding the pattern discovery in network, early works [44], [47], [96] normally preform this task on graphs by identifying frequent patterns from them. Later some efforts have been made to adopt certain topological properties for pattern discovery [27], [31], [60]. Although these techniques are also applicable to AGs, patterns discovered by them are possibly not appropriate as the attribute information in AGs is not considered during the discovery process.

To tackle this problem, certain efforts [19], [95], [103], [104] have been made recently so that patterns can be discovered from AGs in a more appropriate manner. However, the approaches proposed for AGs process the topology information and the attribute information separately while ignoring the association between them. For example, BAGC [95], proposed for the problem of AG clustering, adopts different probabilistic models to quantitatively describe the topology information and the attribute information respectively, and then formulates an optimization problem by involving these models in a unified clustering framework. Therefore, these approaches intend to achieve a balance between the information of topology and that of attribute through

an optimization procedure. However, few of them consider the association between the information of graph topology and that of vertex attributes, which is is yet to be explored.

To better illustrate the significance of considering the association between the information of graph topology and vertex attributes, let us see an example. Assuming that there are two books on the shelf for sale and the keywords related to the content of these two books are also available, there is nothing worth noting except the similarity between these two books in terms of their keywords; but if these two books are co-purchased by the same reader, an intuitive idea coming to mind is to wonder whether these two books are associated or at least some of their keywords are associated. From this example, we believe that it is the existence of edges between vertices that facilitate the understanding of vertex attributes and vice versa.

Therefore, motivated by this intuitive idea, we propose an innovative algorithm to discover association patterns from AGs by disclosing the association between graph topology and vertex attributes. To unify these two kinds of information, association patterns are defined as pairs of attribute values that frequently co-occur in an AG. To demonstrate the ability of such kind of patterns, we apply the association patterns to some of applications of AGs and the experimental results show the promising performance of these patterns.

1.1.2 Approach Innovations

With association patterns discovered from AGs, we are interested in developing specific approaches to solve existing problems whose data can be modeled as AGs. In this thesis, the problems of link prediction and graph clustering have been studied and innovative approaches are then proposed to solve them.

1.1.2.1 Link Analysis

The problem of link analysis we are interested in is to predict links between two vertices through a supervised learning process. In this thesis, we narrow down the scope of AGs by concentrating on PPINs that are more popular in the field of bioinformatics. In this regard, the problem of link analysis we intend to solve for PPINs is to predict PPIs given a confirmed PPIN. To construct the AG from a PPIN, the protein sequence information is used to be the attribute of proteins in the AG.

Previous sequence-based approaches [8], [9], [57], [69], [79] for predicting PPIs are concerned with the sequence segments with the same length and compare the similarity to the distribution of such kind of segments for the task of prediction. However, this consideration cannot thoroughly explore the sequence information as it ignores the association between segments with variable length. Besides, we also take into consideration both the presence and absence of segments to compose the patterns of interest. Therefore, we develop an innovative approach, namely VLSPAD, to address this concern. VLSPAD firstly employs an Apriori-like procedure to identify sequence segments that frequently occur in the protein sequences. Then these sequence segments are assigned to proteins as their attribute values so that an AG can be constructed from the given PPIN. The proposed algorithm of pattern discovery is then applied by VLSPAD so that the associations between segments can be discovered. With these associations, VLSPAD then performs the prediction task for query proteins. The experimental results show that by considering the patterns composed of segments with variable length VLSPAD outperforms some popular sequence-based approaches in terms of accuracy.

1.1.2.2 Graph Clustering

Graph clustering is to identify clusters by grouping related vertices from a graph [77]. The clustering process is much simpler for graphs where no attributes are considered and hence a number of clustering approaches have been developed to identify clusters

by following certain topological properties including but not limited to density [27], [30], connectivity [34], [60] and *k*-clique [12], [82]. Concerning the problem of AG clustering, the foundation of our works is based on the associative patterns identified from AGs. For pairwise vertices, based on the amount and the significance of associative patterns that are found in their attributes, the DOA measure can quantify the degree of association between them, which indicates to what extent these two vertices are likely to be grouped.

Given an AG and the DOA measure, we propose a fuzzy-based clustering approach, namely FC-AG, by combining the topology and attribute information of AGs. First of all, FC-AG preprocesses the attribute information with the use of DOA measure. After that, in order to identify clusters composed of vertices that are densely connected and that are highly associated, FC-AG formulates an optimization problem for the original clustering problem. For each of vertices, the fuzziness of its memberships can be estimated by this optimization problem by considering the topology and attribute information simultaneously. FC-AG then solves the optimization problem in an iterative manner.

However, since FC-AG adopts the fuzzy clustering, the number of clusters has to be determined in advance. Recognizing that there are also many applications where the number of clusters is unknown, we then develop two unsupervised clustering approaches utilizing both the topology and attribute information and they are MCL-AG and CAP-AG. For MCL-AG, the clustering task is performed in several steps: 1) it makes use of DOA measure to weight each of edges in the AG and creates a weighted AG (wAG), 2) based on wAG, it discovers all dense subgraphs in it using markov clustering process, and 3) a partitioning process is applied to each subgraph so that clusters can be identified as partitions consisting of vertices whose attribute values are more closely associated with each other.

In contrast to MCL-AG, CAP-AG solves the problem of AG clustering from an alternative view. As existing works on AG clustering normally concentrate on

grouping vertices that are similar in both topological structures and vertex attributes, such consideration ignores the individual impacts of attributes on the formation of clusters. Hence, we propose a clustering approach using attribute preferences, namely CAP-AG, to identify clusters that are composed of vertices that are similar in a subset of attributes instead of the entire set of attributes and that are densely connected. To do so, CAP-AG employs a likelihood matrix to represent to what extent pairwise vertices are likely to be grouped in the same cluster. For each vertex, CAP-AG also introduces a corresponding preference vector to quantitatively indicate the contribution of each attribute when determining the clustering of this vertex. Then the problem of AG clustering is formulated into an optimization problem. To address it, CAP-AG adopts the strategy of alternatively optimizing the likelihood matrix and preference vectors through an iterative procedure.

1.2 Contributions

The contributions made by this thesis can be presented from four aspects on solving the problem of pattern discovery from AGs, the problem of link analysis and the problem of AG clustering. The highlights are summarized as below.

- Instead of considering the information of topology and vertex attributes independently, we consider them together by discovering associative patterns composed of pairwise attribute values that are frequently observed in adjacent vertices. To do so, we propose an algorithm and then develop several approaches for specific problems based on the associative patterns. The promising performances of the proposed approaches show that associative patterns are of great significance to understand and analyze AGs.

- For the problem of link analysis, one of its specific applications, i.e., the prediction of PPIs, has been studied in the thesis. To solve it, we propose VLSPAD by taking the sequence segments with variable length into

consideration. The experimental results show that VLSPAD outperforms popular sequence-based prediction approaches in terms of accuracy. Therefore, a conclusion can be made that the variable-length segments can improve the accuracy when predicting PPIs.

- Regarding the problem of AG clustering, a fuzzy-based approach FC-AG is proposed. For each of vertices, the fuzziness of memberships is estimated by combining associative patterns and dense link structures. Furthermore, benefited from the adoption of fuzziness in the memberships of clusters for vertices, FC-AG is also capable of discovering overlapping clusters.

- In addition to FC-AG, two unsupervised clustering approaches MCL-AG and CAP-AG have been developed to solve the problem of AG clustering with unknown number of clusters. Both of MCL-AG and CAP-AG simultaneously consider the topology information and the attribute information with the use of associative patterns. The experiment results show the efficiency of MCL-AG and CAP-AG.

1.3 Thesis Organization

The remaining content of this thesis is organized as follows.

- In Chapter 2, the details of discovering associative patterns from AGs are introduced and this chapter also covers the mathematical preliminaries for the following chapters.

- In Chapter 3, we elaborately introduce of procedure of how VLSPAD predicts PPIs from PPINs with variable-length sequence segments. To evaluate the performance of VLSPAD, we apply it to the PPINs extracted from different

species and compare it with two popular sequence-based approaches. The experimental results and analysis are also given.

- In Chapter 4, we introduce a fuzzy-based clustering approach FC-AG. In the experiments, we first use synthetic datasets to demonstrate the advantage of FC-AG in terms of efficiency; then we apply FC-AG to the applications of document classification and social community detection and present an in-depth analysis to the results.

- In Chapter 5, the two unsupervised clustering approaches MCL-AG and CAP-AG are introduced. To evaluate their performances, we apply these two clustering approaches to identify protein complexes from PPINs with protein attributes. In addition, we also select some protein complexes identified by these two approaches to demonstrate the respective advantages of MCL-AG and CAP-AG.

- In Chapter 6, a conclusion about the works in this thesis is presented. Furthermore, we discuss several aspects of how to extend our research works as well as improving the research outcomes as future works.

Chapter 2 Association Pattern Discovery in AGs

2.1 Overview

Existing approaches proposed to solve the problems of AG process the topology information and the attribute information separately and hence ignore the associations we are interested in. Taking distance-based approaches for the problem of AG clustering as an example, the use of distance measures is to quantitatively assess to what extent the two adjacent vertices are similar in terms of their attributes and hence the topological information of AG is not considered by distance measures. Since it is important to take the relationship between the topological information and the attribute information into consideration when solving the problems of AGs, few approaches can do so.

Recall that the existence of edges between vertices facilitates to understand vertex attributes and vice versa, we have reason to believe that associations are possibly existed between the information of graph topology and that of vertex attributes. The difficulty lying here is how to identify patterns representing such kind of associations by considering both the information of graph topology and vertex attributes. To solve it, we consider this problem from an alternative view as described below.

Assuming that there are two attribute values regardless of whether they are from the same attribute, the co-occurrence of them means that they are found in the corresponding attributes of adjacent vertices respectively. It is intuitive for us to consider that these two values, to some extent, represent a certain association between graph topology and vertex attributes if they are observed to frequently co-occur in adjacent vertices. In this regard, the co-occurrence of the two attribute values we

concern is eligible to compose the pattern of association. Regarding the use of the topological information and the attribute information, the attribute values in the association pattern involve the attribute information while the co-occurrences of these two attribute values are counted through edges.

Therefore, if we have such a way that all pairwise attribute values that frequently co-occur in the AG can be discovered as the patterns of this AG, the problems related to AG can be addressed in a more natural and appropriate manner. Therefore, motivated by this idea, an innovative algorithm is introduced in this thesis to discover patterns composed of pairwise attribute values frequently co-occurred.

The rest of Chapter 2 is organized as follows. The related works of pattern discovery in network data are presented in Section 2.2. Section 2.3 introduces the mathematical preliminaries with respect to the model of AG. A complete procedure of pattern discovery is presented in Section 2.4. A conclusion is given in Section 2.5.

2.2 Related Works

2.2.1 Frequent Pattern Discovery

The algorithms developed for the discovery of frequent substructures are normally applied to labeled graphs. Though labeled graphs are also a kind of AG, we still prefer to use labeled graphs when introducing related works in order to avoid any ambiguity. According to the nature of input graphs, we can classify the related algorithms into two groups, one requires the input as a single large graph and the other is to accept a set of small graphs as input. The popular algorithms respectively from these two groups are introduced in the following part.

Ketkar et al. [44] describe the SUBDUE system to discover frequent substructures from the input graph based on the minimum description length (MDL) principle. The

frequency concept used in SUBDUE is not strictly in accordance with the traditional concept that counts on the appearances. Instead SUBDUE intends to discover substructures that better serve for compressing the input graph. To achieve the target, SUBDUE measures the compression ability of substructures with the use of MDL. Regarding the process, SUBDUE begins with substructures matching unique labels in the graph. An iteration process is applied. At each iteration SUBDUE selects the substructures with the best MDL scores and expand them by adding one edge at all possible ways for the next iteration. The result list of SUBDUE reserves the best substructures discovered during each iteration. Regarding the performance of SUBDUE, there are two factors affecting the compression ability of substructures as one is the number of appearances and the other is the size. Therefore, some substructures in the final result set may not appear frequently in the original graph but have a larger size. This could negatively affect the quality of discovered substructures in terms of frequency.

FSG [47] is proposed to extract frequent substructures from a set of small graphs by following level-by-level expansion adopted by Apriori. As graphs are more sophisticated than transaction records, FSG devotes much effort on the problem of how to generate (k+1)-candidates from the frequent set of k-subgraphs. To address this problem, FSG makes use of subgraph isomorphism to examine whether a pair of frequent k-subgraphs shares the same (k-1)-core structure. If they are, FSG then performs the join operation to generate a set of (k+1)-candidates from them through edge expansion. After obtaining all (k+1)-candidates by scanning the frequent k-subgraphs, FSG makes use of the downward closure property of Apriori to remove unqualified (k+1)-candidates, thus retaining the remaining ones as the frequent set of (k+1)-subgraphs. FSG is completed when there is no qualified candidate left. Because of the isomorphism in graph topology, a lot of effort is made to generate candidates with the use of canonical labeling and therefore the efficiency is also affected.

To overcome the inefficiency of FSG, Yan et al. [96] develop gSpan to discover frequent subgraphs from the graph dataset without generating candidates. gSpan

adopts the depth-first search (DFS) strategy to discover frequent subgraphs. To accelerate the search speed, gSpan introduces a new canonical labeling method for graphs. For each of input graph dataset, gSpan constructs all DFS trees of it, then encodes these trees with DFS code scheming, and takes the minimum DFS code as the canonical label of this graph. After obtaining all canonical labels for the input graph datasets, gSpan starts to build DFS code tree. First of all, gSpan picks all frequent 1-edge graphs out of the graph dataset, and sorts these graphs in DFS lexicographic order. Then, gSpan iteratively searches all descendants of each 1-edge graph in this sorted graph list and retains the frequent ones to extend the DFS code tree. At the end of each iteration, gSpan shrinks graphs in the graph dataset by removing the edge selected as the root of this iteration and as result, the removed edge will not appear in the following search. When all 1-edge graphs are searched, a complete DFS code tree has been constructed and frequent subgraphs can be extracted from this tree.

2.2.2 Topological Pattern Discovery

Unlike frequent pattern discovery techniques, topology pattern discovery techniques perform their tasks by following certain topological properties, such as density, betweenness centrality and minimum cut tree.

Markov cluster algorithm (MCL) [27], [90] is proposed to detect dense regions in sparse graphs and the idea behind MCL is that random walks with a specific length have a higher probability for paths of which the beginning and the ending are in the same dense region than for other paths. To implement the idea, MCL first maps a graph onto a Markov matrix. Then two operators, i.e., expansion and inflation, are introduced to simulate the random walks originated from the initial Markov matrix. In particular, the expansion operator allows nodes to meet new neighbors while inflation is to boost the promotion of favored neighbors as well as the demotion of less favored neighbors. By iteratively applying expansion and inflation, this stochastic process conducted by MCL is soon converged to a stable status, and the connected components after convergence are considered as clusters discovered. However, if the value of

inflation parameter is selected improperly, the performance of MCL may appear to generate many small clusters with two or three vertices because of over-boosting.

Newman et al. [60] provide another possible insight of discovering dense regions in the given graph by following a divisive manner. The purpose of betweenness is to measure edges so that edges lying between communities are favored and those insider communities are disfavored. After obtaining betweenness scores for all edges, the edge with the highest score is then removed from the network and then recalculates the betweenness scores for the remaining edges. By iteratively repeating these two steps, the whole process is completed when there is no edge to be removed. Based on the removal order of edges, a hierarchical dendrogram is readily available for finding a proper division of the given graph. Regarding the goodness of a specific division, Newman et al. also introduce the modularity measure to seek for the best division.

Flake et al. [31] explore the feasibility of performing graph clustering based on the minimum cuts with the given graph. However, since simple minimum cuts cannot guarantee the quality of clustering, an artificial node is introduced to control the quality by providing a lower bound on the expansion of the produced clusters and an upper bound on the edge weights between each cluster and the rest of the graph. After the introduction of an artificial vertex, each vertex in the original graph is connected to this artificial vertex with edge of a constant weight, and thus an expanded graph is obtained. Then the minimum-cut tree is constructed for the expanded graph. Finally by removing the artificial vertex from the minimum-cut tree, all connected components are reckoned as the clusters of the input graph. A problem with respect to the introduction of an artificial root vertex is that since the expanded graph is not the same as the original one in terms of the topology, the generated clusters may not exactly reveal the inherent structure of the original graph

2.3 Mathematical Preliminaries

Since an AG is composed of vertices, edges and vertex attributes, we represent the AG as a 3-element tuple $\mathrm{AG} = \{\mathrm{V}, \mathrm{E}, \Lambda\}$, where $\mathrm{V} = \{v_i\}(1 \le \mathrm{i} \le n_{\mathrm{V}})$ is a set of n_{V} vertices, $\mathrm{E} = \{e_{ij}\}$ denotes all the n_{E} edges that represent the information of graph topology, and $\Lambda = \{\Lambda_m\}$ $(1 \le m \le n_{\Lambda})$ is a set of n_{Λ} attributes that are available to be associated with each of vertices in V. Two vertices v_i and v_j are adjacent vertices in the AG if $e_{ij} \in \mathrm{E}$ and they are incident to e_{ij}. Two edges are adjacent if they share a common vertex. Note that since the case of self-loop is not considered for each of vertices, we have $i \neq j$ for each $e_{ij} \in \mathrm{E}$.

Concerning an attribute $\Lambda_m \in \Lambda (1 \le m \le n_{\Lambda})$, we define the domain $dom(\Lambda_m)$ as a set of possible values that can be taken by Λ_m and the total number of values in $dom(\Lambda_m)$ is n_{Λ_m}. Given v_i, the attribute Λ_m of v_i is denoted as Λ_m^i. For the number of values that are taken by Λ_m^i, unlike previous works normally assuming that each attribute can only take a single value, we allow Λ_m^i to take a set of values denoted as $\mathrm{Val}_m^i = \{val_{mp}^i \mid val_{mp}^i \in dom(\Lambda_m)\}$ and use $\left|\mathrm{Val}_m^i\right|$ to represent the number of attribute values associated with Λ_m^i. If Λ_m^i has no values, we simply set $\mathrm{Val}_m^i = \phi$. For any two attributes of the same vertex, say Λ_m^i and Λ_n^i $(m \neq n)$, it is possible that $\left|\mathrm{Val}_m^i\right| \neq \left|\mathrm{Val}_n^i\right|$. For the same attribute of different vertices, say Λ_m^i and Λ_m^j $(i \neq j)$, it is also possible that $\left|\mathrm{Val}_m^i\right| \neq \left|\mathrm{Val}_m^j\right|$.

To better illustrate the mathematical notations introduced above, we give an example of AG with these notations in Figure 1. Regarding the AG in Figure 1, there are five vertices and eight edges. For each vertex in this AG, total n_{Λ} attributes are associated

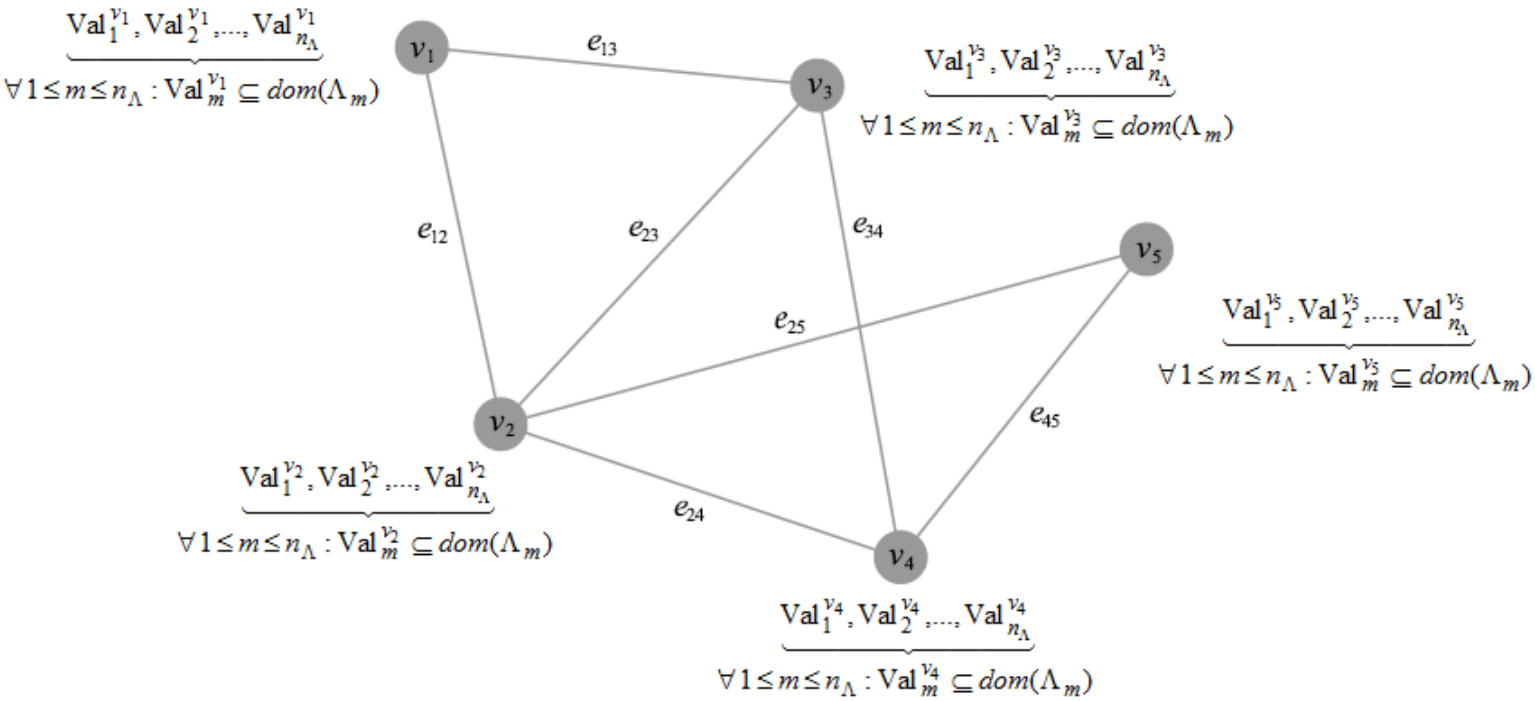

Figure 1. An example of mathematical notations used in an AG. In this graph, each of vertices is associated with several attributes and for each attribute vertices can take more than one values.

with it. More specifically, taking an arbitrary attribute $\Lambda_m \in \Lambda$ of v_3, i.e., Λ_m^3, as an example, a value set denoted as Val_m^3 is to indicate the attribute values assigned to Λ_m^3 and all values in Val_m^3 belong to $dom(\Lambda_m)$. Regarding the co-occurrence of two attribute values, if we have $val_{mp} \in dom(\Lambda_m)$ and $val_{nq} \in dom(\Lambda_n)$, val_{mp} and val_{nq} co-occur through e_{24} if $val_{mp} \in \mathrm{Val}_m^2$ and $val_{nq} \in \mathrm{Val}_n^4$, or $val_{mp} \in \mathrm{Val}_m^4$ and $val_{nq} \in \mathrm{Val}_n^2$.

2.4 Methodology

2.4.1 Discovering Associative Patterns

As mentioned in 1.1.1, the associative pattern is represented by pairwise attribute values that frequently co-occur in adjacent vertices. That is to say, for two attribute

values $val_{mp} \in dom(\Lambda_m)$ and $val_{nq} \in dom(\Lambda_n)$, if the exact number of co-occurrences of val_{mp} and val_{nq} is significantly larger than the expected one, we reckon that val_{mp} and val_{nq} co-occur frequently enough. Hence, val_{mp} and val_{nq} are associated and the co-occurrence of val_{mp} and val_{nq} is defined as the pattern of association.

Let us first consider a simple problem that is to assess the co-occurrences of val_{mp} and val_{nq} given an AG. Assuming that $o_{val_{mp},val_{nq}}$ is the number of pairs of adjacent vertices that have val_{mp} and val_{nq} respectively, $o_{val_{mp}}$ is the number of pairs of adjacent vertices either of which has val_{mp} and $o_{val_{nq}}$ is the number of pairs of adjacent vertices either of which has val_{nq}, we can perform this verification according the inequation below,

$$diff\,(val_{mp}, val_{nq}) = p(val_{mp}, val_{nq}) - p(val_{mp})\,p(val_{nq}) > 0 \tag{1}$$

where

$$p(val_{mp}, val_{nq}) = o_{val_{mp},val_{nq}} \Big/ n_{\mathrm{E}}\,, \tag{2}$$

$$p(val_{mp}) = o_{val_{mp}} \Big/ n_{\mathrm{E}}\,, \tag{3}$$

$$p(val_{nq}) = o_{val_{nq}} \Big/ n_{\mathrm{E}}\,. \tag{4}$$

From (2)–(4), $p(val_{mp}, val_{nq})$ is the joint probability of val_{mp} and val_{nq}, $p(val_{mp})$ is the priori probability of val_{mp}, and $p(val_{nq})$ is the priori probability of val_{nq}.

In (1), the product of $p(val_{mp})$ and $p(val_{mp})$ denotes the expected probability of the co-occurrence of val_{mp} and val_{nq} in the AG under null hypothesis. Obviously, if $diff\,(val_{mp}, val_{nq}) > 0$, it can be inferred that $o_{val_{mp},val_{nq}}$ is larger than its expected value. However, the result of $diff\,(val_{mp}, val_{nq})$ is subject to the magnitudes of $o_{val_{mp},val_{nq}}$, $o_{val_{mp}}$ and $o_{val_{nq}}$ according to (1). Therefore, to avoid the bias resulted

from the difference in magnitude, we rewrite the definition of $diff\,(val_{mp}, val_{nq})$ as below,

$$diff\,(val_{mp}, val_{nq}) = \frac{p(val_{mp}, val_{nq}) - p(val_{mp})\,p(val_{nq})}{\sqrt{\frac{p(val_{mp})\,p(val_{nq})}{n_{\Sigma}}(1 - p(val_{mp}))(1 - p(val_{nq}))}}. \qquad (5)$$

It has been pointed out by [15], [37] that $diff\,(val_{mp}, val_{nq})$ defined by (5) follows a normal distribution and hence the result from (5) is more reliable than that from (1). Hence, to avoid ambiguity, the definition of (5) is adopted to compute $diff\,(val_{mp}, val_{nq})$ afterwards.

However, with (5), we can only reach the conclusion that the co-occurrences of val_{mp} and val_{nq} are more than expected. To further investigate whether $o_{val_{mp}, val_{nq}}$ is large enough to be considered as frequent, we make the decision based on **Theorem 1** as below.

Theorem 1: $o_{val_{mp}, val_{nq}}$ is frequent at a 95% confidence interval if $diff\,(val_{mp}, val_{nq}) \geq 1.96$. If $o_{val_{mp}, val_{nq}}$ is frequent, val_{mp} and val_{nq} are associated and their co-occurrences compose an association pattern representing the association relationship between them.

2.4.2 Weighting Associative Patterns

Though whether there is a pattern existed between val_{mp} and val_{nq} can be determined by **Theorem 1**, the weight of such pattern should not be considered as the same for all patterns discovered. If a pattern is verified, its weight is to indicate the strength of the association between the attribute values involved. It is for this reason that we make use of the weight of evidence [91] to assess the weight of pattern from the viewpoint of decrease in uncertainty.

Firstly, the amount of evidence provided by the presence of val_{mp} for val_{nq} can be estimated by mutual information as below,

$$I(val_{nq} : val_{mp}) = \log\left(\frac{p(val_{nq} \mid val_{mp})}{p(val_{nq})} \right) \tag{6}$$

where $p(val_{nq} \mid val_{mp}) = p(val_{mp}, val_{nq}) / p(val_{mp})$ is the conditional probability of val_{nq} found in a vertex given that val_{mp} is found in its adjacent vertex. From (6), we find that the value of $I(val_{nq} : val_{mp})$ is only positive when $p(val_{nq} \mid val_{mp}) > p(val_{nq})$. In other words, the mutual information defined by (6) shows the decrease in uncertainty about the presence of val_{nq} in a vertex when val_{mp} is found in its adjacent vertex.

Given the presence of val_{mp}, the difference in mutual information when val_{nq} is found and when val_{nq} is not found is an estimation of to what extent val_{nq} is likely to co-occur with val_{mp} in adjacent vertices respectively. Assuming that $\overline{val_{nq}}$ denotes the situation that val_{nq} is not found, such difference, denoted by $WOE(val_{nq} \mid val_{mp})$, is to measure the evidence provided by the observation of val_{mp} in a vertex in favor of its adjacent vertex taking on the value val_{nq}, and its formula is

$$WOE(val_{nq} \mid val_{mp}) = I(val_{nq} : val_{mp}) - I(\overline{val_{nq}} : val_{mp}) \tag{7}$$

Substituting (6) for both $I(val_{nq} : val_{mp})$ and $I(\overline{val_{nq}} : val_{mp})$ in (7), we obtain (8) after some algebra operations.

$$WOE(val_{nq} \mid val_{mp}) = \log\left(\frac{p(val_{mp}, val_{nq})(1 - p(val_{nq}))}{p(val_{nq})(1 - p(val_{mp}, val_{nq}))} \right) \tag{8}$$

However, since $WOE(val_{nq} \mid val_{mp})$ and $WOE(val_{mp} \mid val_{nq})$ are different according to (8), we take the average value of them as the strength of the relationship between val_{mp} and val_{nq} and the definition is given in (9).

$$weight\ (val_{mp}, val_{nq}) = \frac{WOE(val_{nq} \mid val_{mp}) + WOE(val_{mp} \mid val_{nq})}{2} \tag{9}$$

If val_{mp} and val_{nq} are not associated as indicated by a value of $diff\ (val_{mp}, val_{nq})$ smaller than 1.96, we simply assign 0 to $weight\ (val_{mp}, val_{nq})$. Therefore, a full description about $weight\ (val_{mp}, val_{nq})$ is presented with (10).

$$weight\ (val_{mp}, val_{nq}) = \begin{cases} 0, & diff\ (val_{mp}, val_{nq}) < 1.96 \\ \dfrac{WOE(val_{nq} \mid val_{mp}) + WOE(val_{mp} \mid val_{nq})}{2}, & \text{otherwise} \end{cases} \tag{10}$$

Hence, for two vertices, a Degree of Association (DOA) measure is introduced with (11) so that the association between them can be quantified based on the amount and the significance of associative patterns found in their attributes.

$$DOA(v_i, v_j) = \frac{\sum_{p=1}^{n_\Lambda} \sum_{q=1}^{n_\Lambda} weight\ (val_{ip}, val_{jq})}{{n_\Lambda}^2} \tag{11}$$

2.5 Conclusion

In this chapter, we have introduced the algorithm of discovering patterns from an AG given. Generally speaking, patterns of interest in this thesis are composed of pairwise attribute values that co-occur frequently. In this regard, both the information of graph topology and vertex attributes are involved to indicate these patterns. To discover such kind of patterns, the proposed algorithm adopts some concepts in probability theory

that can be used to determine whether the difference between the exact number of co-occurrences and the expected one is significant enough. Furthermore, the proposed algorithm also weights the patterns discovered based on information theory. Based on associative patterns, we also introduce the DOA measure to weight the degree of association between vertices.

To evaluate the reasonability behind associative patterns as well as the DOA measure, we apply these patterns to solve specific problems of AGs. Regarding the role of this chapter, this chapter has to be considered as the fundamental basis of Chapters 3-5, as all the approaches proposed in these chapters make use of these patterns to solve the specific problems related to AGs.

Chapter 3 Link Analysis for Predicting PPIs from PPINs

3.1 Introduction

When performing their functions, proteins seldom do so individually. Instead, they perform their functions by interacting with each other as a whole [65]. Hence, if the interactions among proteins can be predicted more accurately, we may be able to better discover unknown functions of a protein based on the known functions of those proteins that it interacts with. In addition, we may also be able to better understand the molecular mechanisms of many biological processes such as DNA regulation, cell signaling, assembly of protein complexes, etc. Given all such possible applications, various attempts have been made to develop techniques to effectively predict *protein-protein interactions* (PPIs).

For example, different high-throughput techniques, such as the two-hybrid systems [22], [41], mass spectrometry [39], [89], and microarray analysis [85], have been developed for systematic and large-scale identification of PPIs. Though more efficient than the low throughput techniques, these high throughput techniques may only predict PPIs with relatively high false-positive rates [74]. Their coverage of PPIs are also usually rather incomplete [64].

For even more effective and efficient PPI prediction, there have recently been some attempts to make use of computational techniques. Depending on the information that they have to rely on to perform their tasks, these techniques can be divided into three types. The first type consists of techniques that make use of genomic information when performing their tasks and they are therefore considered as adopting a genomic approach for PPI prediction. Techniques based on the genomic approach looks for

relevant biological evidence. Examples of such techniques include those that are based on gene fusion [29], the conversion of gene-order [22], and the calculation of prior probabilities of genomic features between interacting proteins [42].

As it is believed that proteins that co-evolve are more likely to interact with each other, the second type of computational techniques developed for PPI prediction performs the task by relying on known evolutionary information observed between interacting proteins. Techniques that adopt such an approach compare, between proteins, various evolutionary descriptors such as phylogenetic profiles [67], domain knowledge of proteins [18], [43] and topological properties of proteins in PPI networks [99], in terms of homolog so that the extent the proteins of interest co-evolve with each other can be quantitatively evaluated for PPI prediction.

The last type of computational techniques for PPI prediction makes use of the information embedded in protein sequences to determine if proteins interact with each other. It is for this reason that they are considered as taking a sequence-based approach to PPI prediction. Such a sequence-based approach is becoming more and more popular as information relating to protein sequences is more readily available nowadays. In deciding if two proteins interact, it should be noted that, other than protein sequence information, many sequence-based techniques [9], [57] also require that additional information about the proteins, such as residue properties and gene ontology, etc., has to be known. More recent techniques, however, tend to focus only on sequence information. One such example is provided by [8] where a pairwise kernel function is introduced to measure the similarity between pairs of proteins based on the similarities of the sequences alone. Another example can be found in [79] where properties related to the symmetry of sequences are used with an s-kernel function when predicting PPIs. Pitre et al. [69] develop the PIPE algorithm by following the idea that PPIs can be predicted by measuring how frequently some protein sequence segments occur together. More recently, there has also been some attempt [101] to use position-specific statistics from protein sequences to predict PPIs.

In general, the prediction algorithms that adopt a sequence-based approach attempt to discover patterns in protein sequences that can be used to determine how likely two proteins may interact with each other. The patterns that they look for are known as *k*-mers, which are typically composed of amino acid sequence segments that are of length *k*. An example of how these sequence segments can be used to predict PPIs can be found from Ben-Hur and Noble [8] where all combinations of 3-mers are used to construct a feature vector for a given protein sequence so that a *support vector machine* (SVM) can be trained to distinguish between interacting and non-interacting proteins.

Though promising, the *k*-mer patterns that these sequence-based algorithms look for are of fixed-length, i.e., *k*. For example, *3*-mers are considered by SVM-based methods [8], [79] and therefore all patterns to compose the feature vectors of proteins are of fixed-length (i.e., 3). For the case of the PIPE algorithm, *20*-mers are considered and the patterns are all of length 20. For PPIevo, only single amino acids in protein sequences are considered and hence, the patterns are all *1*-mer!

Therefore, when predicting PPIs, the performances of existing sequence-based computational algorithms are constrained by the fixed-length-only patterns that they consider. We believe that, if an algorithm can be developed to take into consideration sequence segments of different length at the same time, the accuracy for PPI prediction can be further improved. With this objective in mind, we have developed a *variable-length associative sequential pattern discovery* (VLASPD) algorithm. The VLASPD algorithm is able to discover variable-length patterns for PPI prediction.

Given a database of protein sequences consisting of pairs of proteins that are known to interact and those that are known not to interact with each other, VLASPD is able to discover sequence segments of different length that may provide evidence to support or refute the existence of an interaction relationship between two proteins. For example, let us consider the two protein sequences in Figure 2. In the two sequences, the segments highlighted in gray are not of the same length. If they are found to appear significantly more frequently than expected among interacting proteins, we can argue

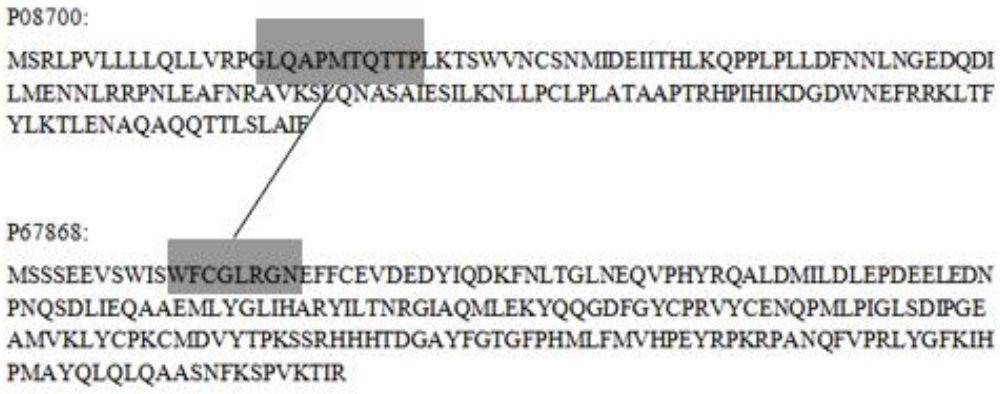

Figure 2. The sequence information of two interacting proteins P08700 and P67868. The two segments highlighted with gray color compose the pattern of interest.

that their existence provide evidence supporting that the proteins P08700 and P67868 interact with each other.

Other than its ability to take into consideration the presence of frequently occurring variable-length sequence segments, VLASPD also takes into consideration how the absence of sequence segments may be used to decide if a protein pair may interact with each other. For instance, for the case of the protein sequences shown in Figure 2, the fact that a 3-mer segment, such as *RHT*, being absent in a protein when another segment, such as *WFCGLRGN*, being present in the other protein may provide evidence to support the existence of interaction relationship between the two proteins, P08700 and P67868!

To do so, VLASPD first searches the given database to identify variable-length sequence segments that occur frequently. These sequence segments are referred to as *frequent sequence segments* (FSSs). The different combinations of the presence and absence of these FSSs form different *associative sequential patterns* (ASPs). The ASPs that occur significantly differently from what are expected among proteins in the database are then identified and they are referred to as *significant associative sequential patterns* (SASPs). These SASPs are considered as providing evidence to support or refute the existence or non-existence of interaction relationship between

two proteins. Using an information-theoretic measure, the amount of such evidence is then computed for all SASPs, making them as *weighted SASPs* (wSASPs). How likely two proteins may interact with each other are then decided by the number and significance of the wSASPs found in them.

In remaining part of this section, we first introduce the background of predicting PPIs in Section 3.1. After that, the related works are given in Section 3.2. Then the details of VLASPD are presented in Section 3.3. For performance, we tested our approach with several other related sequence-based prediction methods and the results are described in Section 3.4. Finally, in Section 3.5, we give a summarization of our work.

3.2 Related Works

In this section, we present a detailed survey on the related work of PPI prediction. For a clear demonstration, the computational approaches proposed for predicting PPIs are classified into three categories based on the sources of biological information they make use of for the task, and they are genomic approaches, evolutionary approaches and sequence-based approaches. Therefore, the following content in this section will be unfolded from these three categories.

3.2.1 Genomic Approaches

Due to the availability of whole genomic sequencing, it has been pointed out that genes located in genome sequences can hint at the interaction between proteins at a comprehensive level.

Dandekar et al. [22] find that proteins encoded by conversed gene pairs are more likely to interact with each other and such conserved gene pairs are within a low level conservation of gene-order. Therefore, with this observation, the conservation of gene-order can be exploited to help predict PPIs. Though proved to be promising [70], this approach fails to predict PPIs for proteins where the conservation of gene-order is not

found, such as proteins encoded by distantly located genes. Furthermore, this approach may not be applicable to species where gene co-regulation is not imposed at the level of genome structure [55].

Another discovery about the use of genomic information related to the formation of PPI is that pairs of interacting proteins are found to have homologs in another genome where they are fused into a single protein [56]. In this regard, several computational methods [29], [88] are developed to seek for such fusion events in different genomes so that proteins involved in a fusion event are expected to interact with each other. However, the disadvantage of this approach is also obvious as it cannot work with proteins where no fusion events are uncovered through the analysis of genomic sequencing.

In [42], different genomic features, such as messenger RNAcoexpression, co-essentiality and co-localization, are used to quantify the associations between them and PPIs. Based on these quantified associations, Bayesian networks are constructed to predict PPIs when the genomic features of query proteins are given.

3.2.2 Evolutionary Approaches

Evolutionary information discloses the procedure of how proteins evolve across different species. Since proteins that co-evolve are more likely to interact with each other, the similarity in evolutionary information is of potential relevance to the prediction of PPIs as it indicates to what extent the two proteins co-evolve.

Among various evolutionary information, Pazos et al. [66] make use of phylogenetic trees of proteins to indicate PPIs. They propose a distance measure to compute the similarity between the phylogenetic trees of proteins, thus determining whether there is a possible interaction between them. Similar to phylogenetic trees, phylogenetic profiles are also adopted by Pellegrini et al. [67] to predict PPIs. Under the assumption that for two interacting proteins one cannot exist if the other one is lost during

evolution, this co-evolution property is characterized by [67] with the use of phylogenetic profiles of proteins and hence proteins with similar profiles are strongly expected to interact.

Another source of evolutionary information that we can use for the prediction of PPIs is the domain knowledge of proteins. It is believed that proteins are to interact as a result of their interacting domains, and it is for this reason that many computations approaches have been proposed to solve the PPI prediction based on domain knowledge. In [54], domains that interact more often than expected are found to compose the signatures of proteins and these signatures are then used to predict PPIs. Rather than simply resting on frequent interacting domains, Deng et al. [24] apply a Maximum Likelihood Estimation method to identify interacting domains that infers curated PPIs and then with such inferred interacting domains the interactions between proteins can be predicted. Similarly, [18] makes use of random forest of decision trees that are trained by taking into consideration all the proteins domains, thus performing the prediction task. In addition, Kanaan et al. [43] employs a set cover approach to partition pairs of domains so that the desired partitioning can best explain the underlying protein interaction in terms of specificity score. Then with the partitioned pairs of domains, the method of Maximum Specificity Set Cover is introduced to predict potential PPIs.

3.2.3 Sequence-based Approaches

Protein sequences, composed of amino acids, are the primary structures of proteins and the motivation behind the use of protein sequences for predicting PPIs derives from the hypothesis that sequence information may contribute to mediate PPIs.

In general, most of sequence-based approaches take advantage of the learning ability of SVM to perform the task. These SVM-based approaches are distinguished by the definition of feature vector extracted from protein sequence and also by the proposal of kernel function used by SVM.

As the first strike among these SVM-based approaches, Bock and Gough [9] assembly the feature vector for each protein sequence based on a set of residue properties, such as charge, hydrophobicity and surface tension. After transforming these vectors with variable lengths into vectors with a fixed length, Bock and Gough then train several SVMs with different standard kernel functions by taking these new vectors as input and make a prediction based on the average results from trained SVMs for query proteins.

Similar to the approach of Bock and Gough, Martin et al. [57] also make use of sequence information as well as experimental data to compose the feature vector of protein. However, Martin et al. additionally extends the signature descriptor to describe interacting proteins that is much more close to the actual biology of PPI and combine such description in the feature vector. Regarding the kernel function of SVM, Martin et al. adopt the signature product to compute the similarity between pairs of proteins based on their feature vectors.

When compared with previous SVM-based approaches, Benhur and Nobel [8] introduce a pairwise kernel function that measures the similarity between pairs of proteins based on the similarities between individual proteins. To extract the feature vector from sequence information, Benhur and Noble adopt the spectrum vector composed of *k*-mers where $k = 3$.

Later Shen et al. [79] also propose an S-kernel function that is specifically designed for PPIs by considering the symmetry property of PPI. Before assembling feature vectors of proteins, Shen et al. classify the amino acids into 7 classes and hence the number of unique elements in the protein sequence is now reduced to 7. With the new protein sequences, Shen et al. introduce a conjoint triad method to create feature vectors for proteins. When considering the difference of feature vector between [8] and [79], the length of the later one, i.e., 7^3, is much shorter than the former one, i.e., 20^3.

In addition to SVM-based approaches, Pitre et al. [69] propose PIPE to tackle the problem of predicting PPIs from a different angle and the idea behind PIPE is to measure how often pairs of subsequences in the two query proteins co-occur in pairs of proteins that are known to interact. Specifically speaking, PIPE first builds a database composed of known interacting proteins. Then for a pair of query proteins, PIPE chooses any of them to as the first one to process. The selected protein sequence is fragmented into segments with the same length. For each segment, PIPE searches each sequence in the database to see whether this segment is contained in it through a matching measure; if a sequence is found to contain this segment, PIPE adds to a neighbor list associated with this segment the neighbors of this sequence. Once all the segments of the first query protein are associated with neighbor lists, the other query protein is then fragmented in the same way. Regarding the two sets of segments of query proteins, PIPE creates a matrix where columns are to denote the segments of the first query protein and rows are for the segments of the other query proteins. For each cell in this matrix, its value is the number of sequences in the corresponding neighbor list of the column segment and each of counted sequences is found to contain the row segment. With this matrix, PIPE scores the confidence about the hypothesis that the query protein are interacting.

3.3 The Details of VLASPD

3.3.1 Identifying FSSs

Given a database T_{train} consisting of protein pairs that are known to interact with each other and protein pairs that are known to be non-interactive, we define a *k*-mer to be a segment of a protein sequence of length *k*. For each *k*, VLASPD begins work by identifying *k*-mers that occur frequently. These frequent *k*-mers form a set of FSSs,

denoted as $\mathbf{S} = \bigcup_k S_k$ where $S_k = \{ s_{ki} \} (1 \le i \le |S_k|)$ is a subset of all FSSs of length *k*, and $|S_k|$ is the number of such sequence segments.

To efficiently obtain **S** from T_{train}, VLASPD makes use of the well-known apriori algorithm [3]. The apriori algorithm is originally developed to identify frequent itemsets in transaction databases. The reason why the apriori, rather than an exhaustive search, algorithm is used is that only some combinations of *k*-mers are usually found in T_{train}. An exhaustive search algorithm is therefore unnecessary and may be intractable as the size of all possible combinations of *k*-mers can grow exponentially when the value of *k* increases.

To use the apriori algorithm, each protein sequence in T_{train} is considered a transaction and if we considered each amino acid as an item, then a *k*-mer FSS can be identified as a frequent *k*-itemset. Given a minimum support ρ which is a predefined value, then the set of FSSs of length *k*, i.e., S_k, can be identified from the candidate *k*-itemsets and these itemsets can be composed to form candidates for (*k*+1)-mers for the next iteration. Such iterative procedure will end if no more valid FSSs can be identified from the candidates. Therefore, with the apriori algorithm, **S** can be identified effectively.

3.3.2 Forming ASPs

Given two FSSs $s_{ki} \in S_k$ and $s_{lj} \in S_l$ where $1 \le i \le |S_k|$, $1 \le j \le |S_l|$, $S_k \subseteq \mathbf{S}$ and $S_j \subseteq \mathbf{S}$, four ASPs can be formed and these four ASPs are represented as $\langle s_{ki}, s_{lj} \rangle$, $\langle \overline{s_{ki}}, s_{lj} \rangle$, $\langle s_{ki}, \overline{s_{lj}} \rangle$ and $\langle \overline{s_{ki}}, \overline{s_{lj}} \rangle$ respectively as shown in Table 1. To denote the presence of a particular *k*-mer segment, say s_{ki}, we directly use its own notation s_{ki}, Hence, the ASP $\langle s_{ki}, s_{lj} \rangle$ means that if s_{ki} is found in one of the sequences of a protein pair, then s_{lj} is found in the other. Since the absence of a certain sequence segment is considered as providing useful information in deciding if two proteins interact,

VLASPD also attempts to determine if the ASPs formed by the absence of a particular *k*-mer segment in one or both sequences of a protein pairs may actually be significant. To denote the absence of a particular *k*-mer segment, say s_{ki}, we make use of the notation $\overline{s_{ki}}$. The ASP $\langle \overline{s_{ki}}, s_{lj} \rangle$, therefore, carries the meaning that if s_{ki} is not found in one of the sequences of a protein pair, then s_{lj} is found in the other. Similarly, $\langle s_{ki}, \overline{s_{lj}} \rangle$ means that if s_{ki} is found in one of the sequences of a protein pair, then s_{lj} is not found in the other and $\langle \overline{s_{ki}}, \overline{s_{lj}} \rangle$ means that if s_{ki} is not found in one of the sequences of a protein pair, then s_{lj} is also not found in the other.

Table 1. The contingency table between the four combinations of s_i and s_j and the relationships of proteins

	r_{int}	$r_{\overline{int}}$
$\langle s_{ki}, s_{lj} \rangle$	o_{11}	o_{12}
$\langle \overline{s_{ki}}, s_{lj} \rangle$	o_{21}	o_{22}
$\langle s_{ki}, \overline{s_{lj}} \rangle$	o_{31}	o_{32}
$\langle \overline{s_{ki}}, \overline{s_{lj}} \rangle$	o_{41}	o_{42}

3.3.3 Discovering SASPs

The four ASPs represent all possible combinations of the presence and absence of the two FSSs, s_{ki} and s_{lj} in a protein pair. In order to find out if these ASPs may have anything to do with whether or not the two proteins in the pair interact with each other, VLASPD investigates into the statistical significance of each of the ASPs among protein pairs that are known to be interactive represented as r_{int} or those that are known to be non-interactive represented as $r_{\overline{int}}$.

To do so, let us consider the association between $\langle \overline{s_{ki}}, s_{lj} \rangle$ and r_{int}. To confirm such association, we compute the result of $diff(\langle \overline{s_{ki}}, s_{lj} \rangle, r_{int})$ with (5) and then the

association between $\langle \overline{s_{ki}}, s_{lj} \rangle$ and r_{int} can be verified according to **Theorem 1**. Regarding the relevant probabilities, we have

$$p(\langle \overline{s_{ki}}, s_{lj} \rangle, r_{int}) = o_{21}/o_{++} ,$$

$$p(\langle \overline{s_{ki}}, s_{lj} \rangle) = o_{2+}/o_{++} ,$$

$$p(r_{int}) = o_{+1}/o_{++} ,$$

where o_{2+} is the sum of values of cells at the second row, o_{+1} is the sum of values of cells at the first column and o_{++} is the sum of values of all cells in Table 1.

As mentioned in 2.2.1, $diff(\langle \overline{s_{ki}}, s_{lj} \rangle, r_{int})$ is to examine whether $\langle \overline{s_{ki}}, s_{lj} \rangle$ and r_{int} frequently co-occur. That is to say, if $diff(\langle \overline{s_{ki}}, s_{lj} \rangle, r_{int})$ is not less than 1.96, a conclusion can be made that $\langle \overline{s_{ki}}, s_{lj} \rangle$ is frequently observed in the pairs of interacting proteins. In this regard, $\langle \overline{s_{ki}}, s_{lj} \rangle$ is associated with r_{int} and it is a SASP. In this regard, as SASPs provide some evidence in deciding if proteins in a protein pair that possesses such pattern may interact or not interact with each other, the SASPs can help predict PPIs.

3.3.4 Weighting SASPs

To make use of the SASPs in predicting PPIs, the amount of evidence that each SASP provide supporting or refuting the existence or non-existence of interaction relationship has to be quantified. To quantify the amount of evidence, VLASPD makes use of the technique introduced in 2.4.2. Taking $\langle \overline{s_{ki}}, s_{lj} \rangle$ as an example, the weight of

$\langle \overline{s_{ki}}, s_{lj} \rangle$ is equal to $WOE(r_{int} | \langle \overline{s_{ki}}, s_{lj} \rangle)$ as defined by (7). Note that $p(r_{int} | \langle \overline{s_{ki}}, s_{lj} \rangle) = o_{21}/o_{2+}$.

The value of $WOE(r_{int} | \langle \overline{s_{ki}}, s_{lj} \rangle)$ is therefore an estimation of the extent two proteins are likely to interact with each other. When $WOE(r_{int} | \langle \overline{s_{ki}}, s_{lj} \rangle) > 0$, it means that the presence of $\langle \overline{s_{ki}}, s_{lj} \rangle$ in a pair of proteins provides evidence in support of the existence of interaction relationship between them. When $WOE(r_{int} | \langle \overline{s_{ki}}, s_{lj} \rangle) < 0$, it means that the presence of $\langle \overline{s_{ki}}, s_{lj} \rangle$ in a pair of proteins provides against the existence of interaction relationship between them. Hence, VLASPD can weight each SASP so that it becomes a weighted SASP, or wSASP. In the following, we use Γ to denote the complete set of wSASPs, $\Gamma_{int} \subseteq \Gamma$ to denote the set of wSASPs whose weights are positive and $\Gamma_{\overline{int}} \subseteq \Gamma$ to denote the set of wSASPs whose weights are negative.

3.3.5 Predicting PPIs

Given a pair of proteins, to determine if we have sufficient evidence to support or refute the two proteins having an interaction relationship, VLASPD makes use of the set of wSASPs determined above.

If a wSASP $\tau \in \Gamma$ is found in a pair of proteins, then we can say that there is some evidence, of amount, $w(\tau)$, for or against the existence of an interaction relationship between the proteins in the pair. Hence, given a pair of proteins, we can match the protein sequences against all the wSASPs in Γ. The amount of evidence provided by each matched wSASP in the sequences can then be added up to determine a total weight of evidence for or against the proteins having an interaction relationship with each other.

With the total weight of evidence computed, we can determine v_{int} and $v_{\overline{int}}$, which represent, respectively, the sum of the weights of wSASPs that provide evidence for a pair of proteins to have interaction relationship, i.e., r_{int}, or to be non-interacting, i.e., $r_{\overline{int}}$, and they are defined in (12) and (13) below:

$$v_{int} = \frac{\sum_{\tau \in \Gamma' \cap \Gamma_{int}} w(\tau)}{\sum_{\tau \in \Gamma' \cap \Gamma_{int}} w(\tau) - \sum_{\tau \in \Gamma' \cap \Gamma_{\overline{int}}} w(\tau)} \tag{12}$$

$$v_{\overline{int}} = \frac{-\sum_{\tau \in \Gamma' \cap \Gamma_{\overline{int}}} w(\tau)}{\sum_{\tau \in \Gamma' \cap \Gamma_{int}} w(\tau) - \sum_{\tau \in \Gamma' \cap \Gamma_{\overline{int}}} w(\tau)} \tag{13}$$

where $\Gamma' \subseteq \Gamma$ is the set of wSASPs that are found matching the sequences of a given protein pair.

According to (12) and (13), if v_{int} is larger than $v_{\overline{int}}$, it means that there is sufficient evidence to support the existence of interaction relationship for the pair of proteins. Otherwise, the evidence is against the two proteins having an interaction relationship.

So far, we have introduced the complete procedure of how VLASPD works for the task of predicting PPIs. To evaluate the performance of VLASPD, we have conducted experiments on PPI datasets from different species and the experimental results and analysis are given in the next section.

3.4 Experimental Results and Analysis

To evaluate the performance of VLASPD, we tested it with several PPI datasets. The results were compared against those obtained using two different SVM-based methods, SVM(Pairwise Kernel) and SVM(S-Kernel), proposed by [8] and [79] respectively, for predicting PPIs. Both these two SVM-based methods make use of 3-mers to predict PPIs. SVM(Pairwise Kernel) and SVM(S-Kernel) were chosen for performance

evaluation as they are both sequence-based approaches and do not require prior knowledge about proteins when performing their tasks. Furthermore, there are sufficient details presented in related works for a full implementation. When comparing SVM(Pairwise Kernel) and SVM(S-Kernel), the major difference between them is the kernels they use for training a SVM classifier. In particular, SVM(Pairwise Kernel) uses a pairwise kernel function while SVM(S-Kernel) introduces an s-kernel function. In the experiments, their SVM classifiers were trained and tested with the use of the libsvm package [16].

3.4.1 Benchmark Datasets

To avoid any bias in the selection of the training and testing data, the two benchmarking datasets published by [62] were used in our experiments. The first of them involved a set of yeast PPI data obtained from the core set of Saccharomyces Cerevisiae in the *database of interacting proteins* (DIP) [94] and the second was a set of human PPI data obtained from the Human Protein Reference Database [71]. The two datasets made available from these databases therefore include a set of protein pairs that are known to have a interaction relationship between each other and a set of protein pairs obtained by pairing up proteins not found in the these databases so that they were artificially generated as non-interacting protein pairs. In particular, for the yeast dataset, there were 3870 pairs of interacting proteins and 385301 pairs of non-interacting proteins; for the human dataset, there were 17434 pairs of interacting proteins and 1743015 pairs of non-interacting proteins.

Sequence information of human proteins was obtained from the E-utilities service provided by the *national center for biotechnology information* (NCBI) [92] while the sequence information of yeast proteins was obtained from the DIP database as of February 18, 2012 which was the latest version available for our use when we performed our experiments.

3.4.2 Performance Measure

Regarding performance evaluation, it has been pointed out by [62], [63] that different ratios of negative to positive PPIs should be adopted so that the prediction performance can be safely unbiased. Therefore, the two SVM-based methods and VLASPD were tested using six different datasets obtained by combining interacting and non-interacting proteins in different proportions in the yeast and human PPI datasets. Of the six datasets, three of them, Human-ROC1, Human-ROC5, and Human-ROC10, were obtained by mixing interacting and non-interacting proteins in the proportion of 1 to 1, 1 to 5 and 1 to 10 respectively. Similarly, the other three of them, Yeast-ROC1, Yeast-ROC5 and Yeast-ROC10, were obtained by mixing interacting and non-interacting proteins in the proportion of 1 to 1, 1 to 5 and 1 to 10 respectively.

Regarding the parameter settings in the experiments, for SVM(Pairwise Kernel), the parameters were set according to the values recommended by [8], i.e., the penalty parameter was set to be 128 and γ was set to be 0.25. Regarding SVM(S-Kernel), there was no recommended setting for the penalty parameter used by the S-Kernel function. Therefore, the best setting of penalty parameter was determined experimentally by trial-and-error based on ROC performance for SVM(S-Kernel). Regarding the parameter setting of VLASPD, we set the minimum support ρ to be 0.3 when performing the apriori algorithm to obtain $\mathbf{s}$.

For performance comparison, we made use of the ROC curve and the area under it. A ROC curve presents the performance as a trade-off between sensitivity and specificity. It is a curve of true versus false positive rate when a threshold parameter is set. The area under the ROC curve (i.e., AUC) is widely accepted as an index of the accuracy for performance comparison. AUC values are within the range from 0 to 1. The higher an AUC value is, the more accurate is a corresponding algorithm.

3.4.3 Experiment Results

The results of the experiments performed to compare the performance of the VLASPD, SVM(Pairwise Kernel) and SVM(S-Kernel) are given as ROC curves in Figure 3 and the corresponding AUC values are given in Table 2. From Table 2, we noted that VLASPD performed consistently better than the random classifier and the SVM-based methods.

Table 2. The AUC values of VLASPD, SVM(Pairwise Kernel) and SVM(S-Kernel)

	VLASPD	SVM (Pairwise Kernel)	SVM (S-Kernel)
Yeast-ROC1	$0.62^{(1st)}$	$0.51^{(2nd)}$	$0.46^{(3rd)}$
Yeast-ROC5	$0.61^{(1st)}$	$0.54^{(2nd)}$	$0.47^{(3rd)}$
Yeast-ROC10	$0.72^{(1st)}$	$0.62^{(2nd)}$	$0.48^{(3rd)}$
Human-ROC1	$0.68^{(1st)}$	$0.63^{(3rd)}$	$0.67^{(2nd)}$
Human –ROC5	$0.67^{(1st)}$	$0.62^{(3rd)}$	$0.64^{(2nd)}$
Human -ROC10	$0.61^{(2nd)}$	$0.62^{(1st)}$	$0.54^{(3rd)}$

For the SVM-based methods, we noted, based on the AUC curves and AUC values, that their overall performance for the case of the yeast datasets was not so satisfactory when compared to the human datasets. In particular, for Yeast-ROC1 and Yeast-ROC5, the average AUC values of both SVM(Pairwise Kernel) and SVM(S-Kernel) were 0.525 and 0.465 respectively and were both rather low. These results indicated that the 3-mer segments considered by them were not very useful for these SVM-based methods to distinguish between interacting and non-interacting proteins. Unlike SVM-based methods that can only focus on 3-mer segments, VLASPD overcomes this limitation by making use of patterns between sequence segments with variable lengths making it possible for VLASPD to have better performance.

Of the two SVM based methods, we noted that SVM(S-Kernel) did not perform as well as SVM(Pairwise Kernel) with all the yeast datasets, as the corresponding AUC

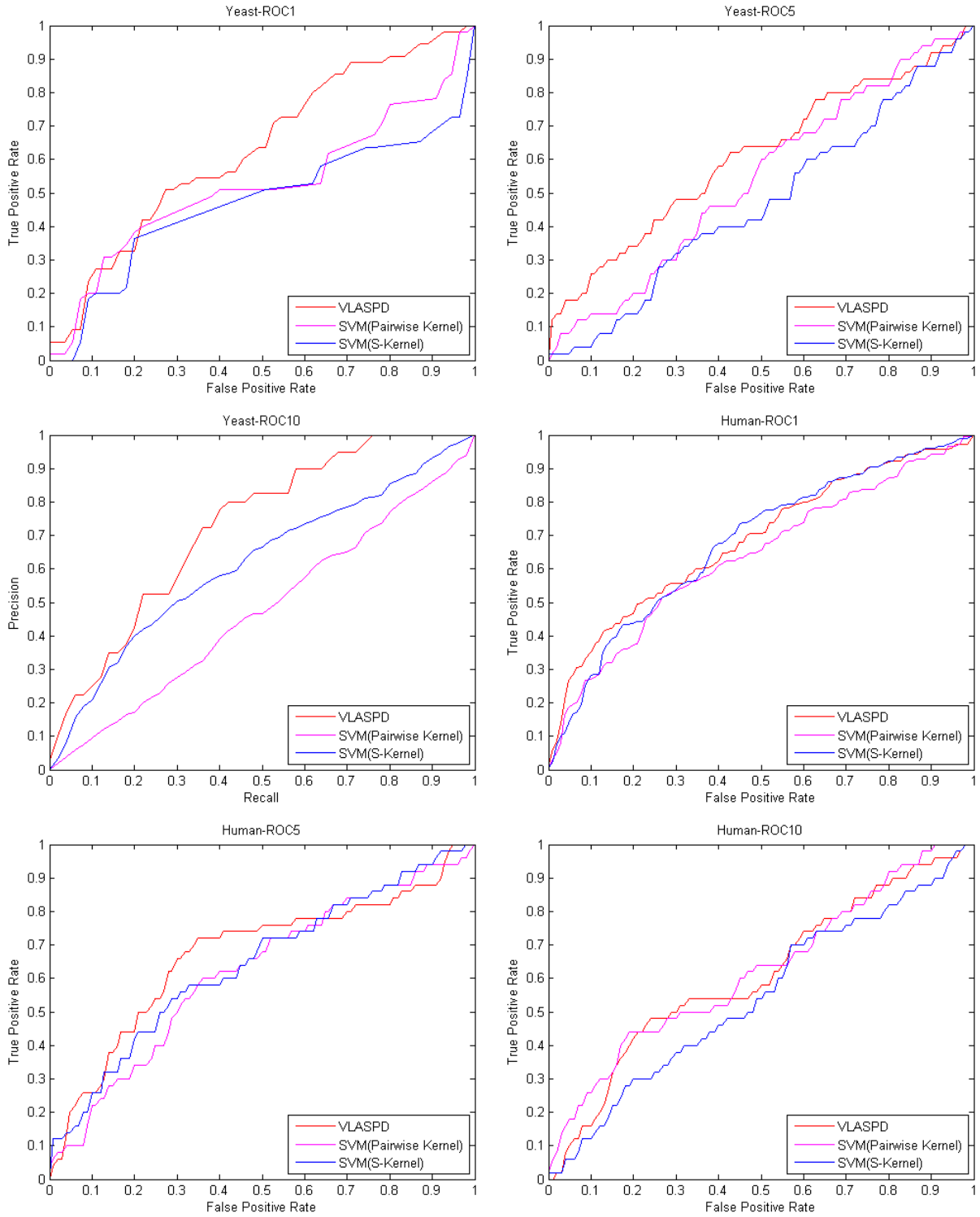

Figure 3. The ROC curves of VLASPD, SVM(Pairwise Kernel) and SVM(S-Kernel) for the datasets of Yeast-ROC1, Yeast-ROC5, Yeast-ROC10, Human-ROC1, Human-ROC5 and Human-ROC10.

values were less than 0.5. In other words, it was better off using a random classifier! Though better than SVM(S-Kernel), we noted that SVM(Pairwise Kernel) was not as effective as VLASPD as the latter was better by 17% on average in terms of AUC value. In this regard, VLASPD outperformed the SVM-based methods in all three cases of the yeast datasets.

Unlike the case with the yeast datasets, the SVM-based methods performed much better with the human datasets. In terms of the average AUC values, SVM(Pairwise Kernel) performed better by 15% when compared with its performance with the yeast datasets. For SVM(Pairwise Kernel), it was better by 22%. In other words, for the prediction of PPIs, 3-mer segments in human datasets are much more relevant than those in the yeast datasets. Since 3-mer segments discovered by these SVM-based methods were also discovered with VLASPD and since VLASPD can also use *k*-mer segments with length other than 3 for PPI prediction, VLASPD can perform better than these fixed-length sequence based methods with the human datasets. However, from Table 2, it is worth mentioning that the average AUC value of VLASPD for the human dataset was more or less the same as the yeast dataset. This indicates that VLASPD is a rather robust algorithm.

For the datasets Human-ROC1 and Human-ROC5, VLASPD performed better. It was the second best with Human-ROC10 but the difference between the first, i.e., SVM(Pairwise Kernel) and the second, i.e., VLASPD, was very small. Hence, when compared with SVM(Pairwise Kernel) and SVM(S-Kernel), VLASPD is less sensitive to the choice of segment length and is therefore more efficient and robust.

For SVM-based methods, it has been pointed out by Batuwita et al. [7] that there is a chance that suboptimal models be constructed by an SVM classifier trained on imbalanced datasets and such models may be biased towards the class that is the majority and in such case, the SVM will perform poorly when predicting minority classes. For this reason, SVM-based methods are very sensitive to changes in the ratio between interacting and non-interacting proteins and this concern has already been

confirmed by the experimental study of Yu et al. [100]. Compared to SVM-based methods, VLASPD is better able to address the concern as it concentrates mainly on the discovering of associations between sequence segments using (5) and (7). Hence, the impact of imbalance between the proportions of interacting and non-interacting proteins can be reduced to an acceptable level and VLASPD's ability of predicting PPIs will therefore not be biased.

3.4.4 Statistical Significances of SASPs

To determine the statistical significance of the SASPs discovered by VLASPD, we adopted a *p*-value test which is a popular statistical significance test commonly used in many applications, such as the identification of protein complexes [40] and social network analysis [28].

For our analysis, SASPs with *p*-values smaller than or equal to the significant thresholds of 0.1 and 0.05 in the training set are considered statistically significant. The experimental results of *p*-value tests are presented in Table 3. It is noted that for each dataset, at least half of the SASPs are significant at both $p \le 0.1$ and $p \le 0.05$. In particular, for Yeast-ROC10, more than three quarters of the SASPs have *p*-values larger than or equal to 0.1. Hence, for proteins with unknown interacting relationship, these SASPs are believed to be able to facilitate prediction. In this regard, VLASPD is preferred as it can identify such significant patterns in an efficient way.

Table 3. The percentage of SASPs that pass the p-value test

	$p \le 0.1$	$p \le 0.05$
Yeast-ROC1	72%	55%
Yeast-ROC5	50%	50%
Yeast-ROC10	77%	62%
Human-ROC1	59%	58%
Human –ROC5	64%	50%

Human -ROC10	64%	52%

Table 4. Top 10 predictions of PPIs identified from respective Yeast and Human datasets

Yeast		**Human**	
PPI	v_{int}	*PPI*	v_{int}
CRYGD and CALB2	**0.91**	**TP53 and SMAD2**	**0.99**
CRYGA and CRYGB	**0.9**	**CREBBP and NKX2-1**	**0.97**
CRYZ and CTAA1	0.9	**CREBBP and TP53**	**0.97**
ELN and CA11	**0.89**	BMP4 and CREBBP	0.97
DCN and CRYGD	**0.89**	CREBBP and THAP1	0.97
DAB2 and CA11	**0.87**	**CREBBP and SMAD2**	**0.96**
ADAM9 and FCGR2B	0.86	**STAT2 and CREBBP**	**0.94**
EPHB4 and PHKA1	**0.84**	**E2F1 and CREBBP**	**0.93**
CRYGB and CALB2	**0.84**	**CSNK1A1 and TP53**	**0.92**
CACNA1B and CAV2	0.84	**HNF4A and CREBBP**	**0.91**

Note: Pairs in bold are known PPIs

In Table 4, we list the 10 PPI predictions with the highest vii for the yeast and human datasets respectively. The entries in bold contains proteins that were previously confirmed to interact with each other. Out of the top 10 predictions in yeast datasets, there were seven such protein pairs and out of the top 10 predictions in human datasets, there were eight such known pairs. The high matching between predicted and known PPIs among the top 10 predictions is an indication that SASPs can be very effective in distinguishing between interacting and non-interacting proteins. Therefore, for proteins whose interaction relationships are as yet unknown, we have reasons to believe that there exists interaction relationship between them but that these relationship might have been missed by laboratory experiments if with large values of v_{int}. Taking the proteins ADAM9 and FCGR2B as an example, since the score of v_{int},

i.e., 0.86, is high, there is a high probability that they do indeed interact with each other.

To try to confirm that this is really the case, we further looked up their gene information in the Gene Ontology (GO) database [14]. We and found that these two proteins perform similar protein binding functions and are both co-located in membrane cells. Hence, considering that the relatively high v_{int} and the fact that their similar GO information, there is relatively strong evidence that there exist interaction between ADAM9 and FCGR2B which might have been missed by previous laboratory experiments.

3.5 Conclusion

In this chapter, we present an algorithm called VLASPD for PPI prediction. Compared to existing sequence-based methods, VLASPD is able to perform its tasks by considering variable-length sequence segments. The proposal of SASPs allows VLASPD to identify associative sequence segments whose occurrences are believed to provide evidence for or against the interaction relationship of proteins. Furthermore, with some concepts in information theory, VLASPD can quantify the amount of such evidence for making accurate PPI prediction. The experimental results show that the use of patterns identified by VLASPD can considerably improve the accuracy of PPI prediction.

Chapter 4 Fuzzy-based AG Clustering

4.1 Overview

For the clustering of networks, the purpose is to group related vertices into the same cluster [77]. However, when applied to complex networks, such clustering procedure requires to consider both of the link and content information simultaneously, which is known to be challenging. Early clustering approaches perform their tasks by only taking the link information into account while failing to consider the content information of networks. In particular, these approaches identify clusters by specifying particular link structures, including, but not limited to, similarity, density, modularity and min-cut/max-flow. For example, Luxburg [53] discussed the details of how to apply spectral clustering to network data so that clusters composed of vertices with similar link structures can be identified; van Dongen [27] and Nepusz et al. [59] proposed different clustering approaches to discover dense clusters based on markov processes and fuzzy clustering respectively; Newman [60] and Brandes et al. [11] introduced various formulations of modularity to identify clusters efficiently; Flake et al. [31] and Gorke et al. [34] utilized minimum-cut trees to perform the clustering task in a hierarchical manner. Recently, Chen and Saad [17] exploited the idea of reordering/blocking matrices in sparse matrix techniques so as to identify dense clusters from sparse network datasets with the use of link information.

Recognizing that only considering link information is insufficient to identify clusters accurately from network data where the content information is also available, many attempts have been made by taking both of these two kinds of information into consideration. Corresponding clustering approaches can be mainly classified into two categories, one is distanced-based and the other is model-based. In the literature of distance-based approaches [19] , [75], [103], [104], networks are normally augmented by adding new edges that represent the content information, then a variety of distance

measures have been designed specifically to compute the similarity between pairwise vertices by considering all relevant edges in the augmented networks so that clusters can be identified based on the similarities. Regarding model-based approaches, generative models (e.g., iTopicModel [84], BAGC [95] and CESNA [97]) and discriminative models (e.g., PCL-DC [98]) have been developed to simulate the generation of network data using different modeling methods and then clusters that maximize the likelihood of the simulation process are considered as the final result. In addition to these two categories, there are also other approaches that combine content and link information for clustering, such as DB-CSC [36] proposed to identify dense regions in the graph as well as in the attribute space using a novel graph transformation.

Though the efficiency of these approaches have been demonstrated, few of them adopt the technique of fuzzy clustering to identify clusters by considering both of content and link information in a natural manner. The main difficulty in developing such a fuzzy-based clustering approach is how to estimate the fuzziness of memberships for each of vertices in a unified clustering framework that involves both content and link information. To solve this problem, we propose a fuzzy-based clustering approach, namely FC-AG[1], based on content relevance and link structure.

For the processing of the content information, as has been pointed out by [98] that the existence of irrelevant attributes in the content information often leads to unsatisfactory clustering performance, FC-AG introduces a relevance measure that targets to filter out irrelevant content information. Regarding the link structure, we intend to identify clusters with dense structures, which have been proved to be useful for a number of applications [12], [19], [40]. Combining content relevance and link structure, a satisfactory clustering result shall be composed of clusters that follow the intuitive properties:

1) Vertices that belong to the same cluster are densely connected;

[1] Fuzzy-based Clustering approach for Attributed Graphs

2) Vertices that belong to the same cluster shall be strongly relevant with each other;
3) Adjacent vertices are more likely to be grouped in the same cluster.

To do so, we formulate the clustering problem into an optimization problem with the purpose of maximizing the degree of consistence between the resultant clusters and the aforementioned intuitive properties. With the optimization problem, the fuzziness of memberships for each of vertices can be estimated. FC-AG then solves this optimization problem in an iterative manner so that the clusters identified are believed to be the best result that satisfies the intuitive properties.

To evaluate the performance of FC-AG, we have conducted extensive experimental studies. In particular, we used synthetic datasets to 1) verify whether the resultant clusters are consistent with the intuitive properties aforementioned, 2) perform the sensitivity tests for parameters used in the optimization problem and analyze the results, and 3) assess the scalability of FC-AG on large-scale datasets. In addition to synthetic datasets, we also applied FC-AG to two practical applications, one was document classification and the other was the identification of social communities. The experiment results have demonstrated the efficiency and scalability of FC-AG.

The rest of this section is organized as follows. In Section 4.1, an overview for the AG clustering is given. Then we present a details literature review in Section 4.2, following which we state the problem of AG clustering that is about to be tackled in Section 4.3. The details of FC-AG are demonstrated in Section 4.4. To analyze the performance of FC-AG, we conducted extensive experiments on synthetic datasets and the results are presented in Section 4.5. The advantage of FC-AG on practical applications has been demonstrated in Section 4.6. Finally, we end the paper with a conclusion in Section 4.8.

4.2 Related Works

Existing AG clustering algorithms are mainly classified into two categories. The first category is composed of distance-based approaches, such as SA-Cluster [103] and its extended version SA-Cluster-Opt [19], and Inc-Cluster [104]; the other category is model-based approaches, such as iTopicModel [84], BAGC [95], CESNA [97], PCL-DC [98] and DB-CSC [36]. The main difference between these two kinds of AG clustering algorithms is that distance-based algorithms rely upon specific distance measures so that the similarity between vertices can be computed while model-based algorithms utilize different probabilistic models to find the clustering with the maximum joint probability.

4.2.1 Distance-based Approaches

Regarding the clustering algorithms that are proposed for AGs where each vertex can be annotated with more than one attributes, the main idea is to weight the edges based on the similarity between two sets of attributes respectively from connecting vertices. In this sense, the AG clustering is converted to the clustering of weighted graphs. Many algorithms have been proposed to address this problem and are introduced as below.

Zhou et al. propose the algorithm SA-Cluster that takes advantage of a unified random walk distance measure to combine both structure similarity and attribute similarity. To address this problem, when given an attributed graph, SA-Cluster initiates an Augmented Attributed Graph (AAG) by taking the input graph as the prototype. Then a set of artificial vertices corresponding to the pattern <attribute, value> is added to this AAG. If a vertex has an attribute with a specific value, it will be connected to the corresponding artificial vertex. The advantage of AAG is that it reveals both the structure information of the original graph and the attribute relationships from its topology. By defining a clustering objective function, SA-Cluster applies the k-

medoids algorithm to group the vertices from the original graph. For each iteration, SA-Cluster automatically adapts the weight values assigned to attributes in order to achieve a convergence upon the objective function. However, there is a bottleneck with respect to SA-Cluster as the computation of random walk distance measure requires a lot time. Therefore, later Zhou et al. introduce Inc-Cluster algorithm to improve the efficiency. Inc-Cluster only calculates the weight changes for attribute edges, thus avoiding extra computation for edges from the original graph. Though the introduction of AAG facilitates the unified measure for both structure and attribute, AAG changes the topology properties of the original graph as some of vertices that are disconnected in the original graph can be linked if they share some attribute values. As a result, after the removal of artificial attribute vertices, clusters discovered may be composed of several separate parts, and therefore the clustering quality is also affected.

4.2.2 Model-based Approaches

In addition to the distance-based approaches, there are also model-based approaches proposed for AG clustering.

Although originally proposed for document clustering, iTopicModel can also be considered as an approach proposed for AG clustering, as it models the documents as graphs where the text information are the attributes of each document in this graph. Following the idea that vertices in the same cluster should have similar distributions of both adjacent vertices and document topics, iTopicModel builds up a model with two layers so that vertices with similar distributions are more likely to be grouped in the same cluster. On the top layer, a multivariate Markov Random Field is proposed to simulate the distribution of topics for each document based on document neworks while the second layer adopts a traditional topic model to model the generation of topics for each document. By combining these two layers, iTopicModel provides a joint distribution function by considering both the topic information and the network information. iTopicModel introduces an optimum solution to maximize the joint probability so that the problem of document clustering can be solved.

Xu et al. consider an alternative view based on a Bayesian probabilistic model combining both structural and attribute information. This model defines a joint probability distribution about the space of all possible combinations of attributed graphs and vertex clustering when vertices, attributes and the number of clusters are given. Therefore, with this model, Xu et al. develop an algorithm BAGC to cluster a given attributed graph based on the computed Bayesian model. BAGC first obtains a posterior probability distribution for all possible clustering results of the given attributed graph, and then takes the clustering with the highest probability as the final result. However, since the generated Bayesian probability model is sensitive to parameters, such as the number of vertices, the set of attributes and the number of clusters, if any of these given parameters is changed, the entire model should be recomputed to fit the new input, which is not efficient for clustering.

To avoid resting on a single source of information in AGs, Yang et al. propose an algorithm of identifying communities from edge structure and node attributes (CESNA), which is capable of detecting overlapping communities. To do so, CESNA uses a probabilistic generative model that combines community memberships, the network topology and vertex attributes. With such a model, CESNA estimates the parameters of this model from the given network and then infers the communities from the model.

Note that many networks contain noise in the graph topology and also that such noise can be identified by the attribute information, Ruan et al. introduce a measure of single strength between two nodes in the network by fusing their link strength with content similarity and then propose PCL-DC to identify communities from AGs. In particular, link strength is estimated based on whether the link is likely to be within a community while the content similarity can be computed through cosine similarity or Jaccard efficient. Then standard community discovery algorithms, such as MCL, are adopted by PCL-DC to perform the clustering task.

To circumvent the constraint of identifying clusters with certain topological properties, Günnemann et al. introduce a new cluster definition that takes the attribute similarity in subspaces and the graph density into account. With this definition, cluster can be detected with arbitrary shape and size. Furthermore, such definition can also avoid redundancy in the result by selecting only the most interesting non-redundant clusters. Based on this cluster model, DB-CSC is developed to identify clusters from AGs by considering both of link and attribute information. DB-CSC intends to identify clusters corresponding to dense regions in the attribute space and also in the graph. To do so, the density of single vertices is based on local neighborhoods taking the attribute similarity in subspaces as well as the graph information into account. After merging all vertices that are in the same dense region, clusters can be identified by DB-CSC.

4.3 Problem Statement

Following the notations introduced in Section 2.3, the clustering problem of network data to be addressed is to partition V into k clusters, i.e., $\mathrm{V} = \bigcup_{f=1}^{k} \mathrm{C}_f$, appropriately. Since the overlapping between any two clusters is possible due to the use of fuzzy clustering, we have $\exists f, g\ (1 \le f, g \le k), \mathrm{C}_f \bigcap \mathrm{C}_g \ne \varnothing$.

For the link information of G, we use a $n_{\mathrm{V}} \times n_{\mathrm{V}}$ matrix $\mathbf{D} = [d_{ij}]\ (1 \le i, j \le n_{\mathrm{v}})$ to represent it. The value of d_{ij} is given in (13).

$$d_{ij} = \begin{cases} 1, & e_{ij} \in \mathrm{E} \\ 0, & others \end{cases} \tag{14}$$

In addition to $\mathbf{D}$, a content relevance matrix $\mathbf{A}$ is also introduced so that the content information can be utilized. Of $\mathbf{A}$, each of cells denotes to what extent the corresponding pairwise vertices are relevant in terms of their attribute information. In particular, for two vertices v_i and v_j, $a_{ij} \in \mathbf{A}$ denotes the degree of relevance

between them and it quantitatively measures how v_i and v_j are likely to be grouped in the same cluster based on the amount and significance of associative patterns found in their attributes. Hence, we have $a_{ij} = DOA(v_i, v_j)$. Note that since the consideration of self-connections of vertices is meaningless for the clustering task, we simply set $a_{ii} = 0$.

For the memberships of each of vertices, we define u_{if} as the value of the membership for v_i with respect to C_f. Obviously u_{if} measures how likely v_i belongs to the cluster C_f. To facilitate the presentation, the vector $\mathbf{u}_i$ is used to represent the membership distribution of v_i over all the k clusters and it is denoted as $\mathbf{u}_i^{\mathrm{T}} = (u_{i1}, u_{i2}, ..., u_{ik})$ where $\sum_{f=1}^{k} u_{if} = 1$. The membership matrix is defined as $\mathbf{U} = (\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_{n_v})^{\mathrm{T}}$.

Given these preliminaries, we will present the details of FC-AG in the next section.

4.4 Methodology

In this section, we first introduce the details of how to formulate a maximum optimization problem for the AG clustering problem by combining content association and link structure. After that, we present the solution to solve this optimization problem and propose FC-AG to implement the solution. Finally, we give an in-depth analysis for recommending the values of parameters involved by the optimization problem.

4.4.1 Problem Formulation

Following the intuitive properties claimed for the clusters identified, we now formulate a constrained optimization problem by combining content relevance and link structure as:

$$\max J(\mathbf{U}) = Tr(\mathbf{U}^{\mathrm{T}}(\partial\,\mathbf{A} + \beta\,\mathbf{D})\mathbf{U}) - \frac{\phi}{2}\|\mathbf{U}\|_F^2 - \frac{\theta}{2}Tr(\mathbf{U}^{\mathrm{T}}\mathbf{S}) \quad s.t.\ \mathbf{U1} = \mathbf{1}, \mathbf{U} \geq 0 \tag{15}$$

where ∂, β, ϕ and θ are within the range [0,1], $\|\mathbf{U}\|_F^2 = Tr(\mathbf{U}^{\mathrm{T}}\mathbf{U})$ is the squared Frobenius norm of $\mathbf{U}$, $\mathbf{1}$ is a column vector with a proper size and each element of $\mathbf{1}$ is 1, and $\mathbf{S} = [s_{if}]$ is a $n_{\mathrm{V}} \times k$ matrix each cell of which is defined as:

$$s_{if} = \sum_{j=1}^{n_{\mathrm{V}}} \sum_{\mathrm{g}=1,\mathrm{g}\neq\mathrm{f}}^{k} d_{ij} u_{jg} \ . \tag{16}$$

So far, the problem of AG clustering is converted into an optimization problem as described by (15), which is composed of three components: a measure of clustering quality, regularizations and constraints. To clarify the eligibility of the optimization problem of (15) in terms of identifying satisfactory clusters, we give a detailed analysis of (15) so that the eligibility can be verified.

Regarding the dense structure of clusters, we constraint our analysis on the part related to $\mathbf{D}$ in the first term of (15) and rewrite it by following the trace expression:

$$Tr(\beta\,\mathbf{U}^{\mathrm{T}}\mathbf{D}\mathbf{U}) = \beta \sum_{i=1}^{n_{\mathrm{V}}} \sum_{j=1}^{n_{\mathrm{V}}} d_{ij}\mathbf{u}_i^{\mathrm{T}}\mathbf{u}_j = \beta \sum_{i=1}^{n_{\mathrm{V}}} \sum_{f=1}^{k} \sum_{j=1}^{n_{\mathrm{V}}} u_{if}\, d_{ij}\, u_{jf} \tag{17}$$

Given the constraint that $\forall i, f : 0 \leq u_{if} \leq 1$, it is not difficult to conclude that if a vertex has a large membership in the cluster C_f and its adjacent vertices also have large memberships for C_f, the value of (15) is to be maximized. Therefore, this

conclusion, to some extent, ensures that vertices in the same cluster are densely connected. Similarly, if we want to maximize (15) by only taking into consideration the part related to $\mathbf{A}$ in the first term of (15), i.e., $Tr(\partial \mathbf{U}^{\mathrm{T}}\mathbf{A}\mathbf{U})$, vertices from the same cluster have to be strongly relevant as indicated by large weights of $\mathbf{A}$. In this regard, the consistence between (15) and the requirement of content relevance for clusters can be verified.

The second term in (15), i.e., $\frac{\phi}{2}\|\mathbf{U}\|_F^2$, is to control the smoothness of memberships for each vertex. The last term in (15), i.e., $\frac{\theta}{2}Tr(\mathbf{U}^{\mathrm{T}}\mathbf{S})$, indicates the penalty of the inconsistence of cluster labels between vertices and their adjacent vertices respectively.

4.4.2 Solution

To solve the maximization problem of (15) with constraints, we consider it as an inequality constrained problem and therefore the method of Lagrange multiplier is applied to eliminate the inequalities.

Assuming that a column vector $\boldsymbol{\lambda}^{\mathrm{T}} = (\lambda_i)\,(1 \le i \le n_{\mathrm{v}})$ and a $n_{\mathrm{V}} \times k$ matrix $\boldsymbol{\Omega} = [\omega_{if}]$ are the langrage multipliers, we can write the Lagrangian formulation of (18) as,

$$\max \Re(\mathbf{U}, \boldsymbol{\lambda}, \boldsymbol{\Omega}) = J(\mathbf{U}) + \boldsymbol{\lambda}(\mathbf{1} - \mathbf{U}\mathbf{1}) + Tr(\mathbf{U}^{\mathrm{T}}\boldsymbol{\Omega}). \tag{18}$$

In (18), both the equality constraint and inequality constraint of (15) are eliminated by the introduction of λ and $\boldsymbol{\Omega}$. The Karush-Kuhn-Tucker (KKT) conditions of (18) are:

$$\partial\Re/\partial\mathbf{U} = 0, \tag{19}$$

$$\partial\Re/\partial\boldsymbol{\lambda} = 0, \tag{20}$$

$$\boldsymbol{\Omega} \circ \mathbf{U} = 0, \tag{21}$$

$$\boldsymbol{\Omega} \geq 0 . \tag{22}$$

Note that $\circ$ denotes the entrywise product of two matrices. If a solution $(\mathbf{U}^*, \boldsymbol{\lambda}^*, \boldsymbol{\Omega}^*)$ satisfies all KKT conditions, $\mathbf{U}^*$ will be considered as the optimal solution of (15).

To find $(\mathbf{U}^*, \boldsymbol{\lambda}^*, \boldsymbol{\Omega}^*)$, we first look at the partial derivative of (19) with respect to u_{if}. After some algebraic manipulations, the memberships of v_i is given by (23).

$$\mathbf{u}_i = \frac{1}{\phi}\left(2\mathbf{h}_i - \theta \mathbf{s}_i^{\mathrm{T}} + \lambda_i \mathbf{1} + \boldsymbol{\omega}_i^{\mathrm{T}}\right) \tag{23}$$

where

$$\mathbf{h}_i^{\mathrm{T}} = (\partial \mathbf{a}_i + \beta \mathbf{d}_i)\,\mathbf{U} \tag{24}$$

and $\mathbf{s}_i$, $\boldsymbol{\omega}_i$, $\mathbf{a}_i$ and $\mathbf{d}_i$ are the i_{th} row vectors of $\mathbf{S}$, $\boldsymbol{\Omega}$, $\mathbf{A}$ and $\mathbf{D}$ respectively. To facilitate the use of $\mathbf{h}_i$, we define the matrix $\mathbf{H} = (\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_{n_v})$. The first term of (29) decides the membership distribution for each v_i in the context of AG and the remaining terms are related the regulations and constraints of (15).

To obtain the value of λ_i in (23), combining the KTT condition (20) and (29) results in the formula of λ_i as expressed below.

$$\lambda_i = \frac{1}{k}\left(\phi + \theta \mathbf{s}_i \mathbf{1} - \boldsymbol{\omega}_i \mathbf{1} - 2\mathbf{h}_i^{\mathrm{T}} \mathbf{1}\right) \tag{25}$$

The last problem left is to derive the value of ω_{if}. According to the KTT conditions related to ω_{if}, ω_{if} is defined as,

$$\omega_{if} = \begin{cases} 0, & u_{if} > 0 \\ 1, & u_{if} = 0 \end{cases} \tag{26}$$

Since (21) is the complementary slackness of the constraint $u_{if} \geq 0$ and (22) is the corresponding sign restriction, the value of ω_{if} is only involved when computing u_{if}. When $u_{if} = 0$, the improving feasible direction of $\Re(\mathbf{U}, \boldsymbol{\lambda}, \boldsymbol{\Omega})$ in terms of u_{if} reaches the boundary and in this regard the solution will not need the value of ω_{if} afterward. However, in order to give ω_{if} a complete description, we simply assign 1 to ω_{if} in case of $u_{if} = 0$.

4.4.3 FC-AG

Since the relationships among $\mathbf{U}$, $\boldsymbol{\lambda}$ and $\boldsymbol{\Omega}$ for the solution of (28) are presented through (23) and (26), we propose FC-AG to find $(\mathbf{U}^*, \boldsymbol{\lambda}^*, \boldsymbol{\Omega}^*)$. An iteration process is adopted by FC-AG to search for a local optimal solution of (25). At the $(l+1)_{th}$ iteration, the previous result of $\mathbf{U}$, i.e., $\mathbf{U}^{(l)}$, will be used to reestimate $\boldsymbol{\lambda}^{(l+1)}$, $\boldsymbol{\Omega}^{(l+1)}$ and $\mathbf{U}^{(l+1)}$ according to (25), (26) and (23) respectively. At the end of the $(l+1)_{th}$ iteration, $\mathbf{U}^{(l+1)}$ will be normalized with (27).

$$u_{if}^{(l+1)} = \begin{cases} \dfrac{u_{if}^{(l+1)}}{\sum_{f \in Z_i^+} u_{if}^{(l+1)}}, & f \in Z_i^+ \\ 0 & f \notin Z_i^+ \end{cases} \tag{27}$$

where $Z_i^+ = \{ f : u_{if} > 0 \}$.

The iteration process will terminate till a convergence, which is defined as the maximum change between $\Re(\mathbf{U}^{(l+1)}, \boldsymbol{\lambda}^{(l+1)}, \boldsymbol{\Omega}^{(l+1)})$ and $\Re(\mathbf{U}^{(l)}, \boldsymbol{\lambda}^{(l)}, \boldsymbol{\Omega}^{(l)})$, or the maximum number of iterations $l_{\max}$ is reached. The final result of $\mathbf{U}$ is considered as $\mathbf{U}^*$. A complete description of the proposed approach is presented in Figure 4.

Algorithm: FC-AG

Input: AG, k, ∂, β, ϕ, θ, ε, $l_{\max}$

Output: the membership matrix $\mathbf{U}^*$

1: obtain the adjacent matrix $\mathbf{D}$ according to (14)
2: obtain the relevance matrix $\mathbf{A}$ according to (11)
3: initialize $\mathbf{U}^{(1)}$ with random nonnegative values and normalize $\mathbf{U}^{(1)}$ with (27)
4: $l = 1$
5: **repeat**
6: update $\boldsymbol{\lambda}^{(l+1)}$ from (25) with $\mathbf{S}^{(l+1)}$, $\boldsymbol{\Omega}^{(l+1)}$ and $\mathbf{H}^{(l+1)}$ computed as (16), (26) and (24) respectively
7: update $\mathbf{U}^{(l+1)}$ with (23)
8: normalize $\mathbf{U}^{(l+1)}$ with (27)
9: $\varepsilon^{(l+1)} = \left| J(\mathbf{U}^{(l+1)}) - J(\mathbf{U}^{(l)}) \right|$
10: $l = l + 1$
11: **until** $\varepsilon^{(l)} \le \varepsilon$ **or** $l > l_{\max}$
12: return $\mathbf{U}^{(l)}$

Figure 4. The Complete Procedure of FC-AG.

4.4.4 Parameter Analysis

As there are several parameters that need to be tuned before applying the proposed approach, we here focus our discussion on how to choose a proper value for ∂, β, θ and ϕ respectively. As these four parameters are involved in determining the membership distribution over k clusters, the selection of their values is of significance to achieve a satisfactory clustering performance in practice.

Regarding ∂ and β, as the two terms $Tr(\mathbf{U}^{\mathrm{T}}\mathbf{D}\mathbf{U})$ and $Tr(\mathbf{U}^{\mathrm{T}}\mathbf{A}\mathbf{U})$ in (15) can have vastly different orders of magnitude due to the size of AG, we use ∂ and β to control

the scaling of these two terms respectively. In particular, if $\partial\mathbf{A}$ is larger than $\beta\mathbf{D}$, $\mathbf{U}^*$ is more likely to be controlled by content relevance between pairwise vertices; if $\partial\mathbf{A}$ is smaller than $\beta\mathbf{D}$, the iteration procedure of the proposed approach will proceed in favor of grouping vertices that are densely connected. Therefore, to obtain a balance result in the sense that both content relevance and link structure are considered equally when FC-AG is searching for $\mathbf{U}^*$, $\partial|a_{avg}|$ and βd_{avg} should have the same order of magnitude given that

$$a_{avg} = \frac{\sum_i \sum_j a_{ij}}{n_V^2}, \tag{28}$$

$$d_{avg} = \frac{\sum_i \sum_j d_{ij}}{n_V^2}. \tag{29}$$

For the proper values of ϕ and θ, since we use them to regulate the smoothness of memberships of vertices, they are not as important as ∂ and β, which are more concerned with the clustering quality. Therefore, for simplicity, we can set the values of ϕ and θ equal to 0.5.

4.4.5 Parallelization of FC-AG

For FC-AG, we implement it in a parallelization manner so that the speed of FC-AG can be boosted by a factor equal to the number of threads. In particular, because $\mathbf{U}^{(l)}$ is fixed when FC-AG updates $\mathbf{U}^{(l+1)}$, updating each $\mathbf{u}_i$ is independent for different vertices. In this regard, updating $\mathbf{U}$ allows for parallelization. Besides, we can also obtain $\mathbf{A}$ in parallel for multiple pairwise attribute values. Obviously the parallelized FC-AG is more efficient when applied to large networks.

4.5 Experiments with Synthetic Data

To evaluate the performance of FC-AG, we have performed extensive experiments with both synthetic and real datasets. In this section, we concentrated on analyzing the experimental results with synthetic datasets so as to demonstrate the advantages of FC-AG in terms of accuracy and scalability.

In the experiments, we compared FC-AG with several state-of-the-art approaches on the task of network clustering, including iTopicModel [84], CESNA [97], Spectral Graph Clustering (SGC) [53] and Fuzzy C-Means (FCM). Briefly speaking, both iTopicModel and CESNA are developed for network data using both link and content information; SGC is one of the few approaches that can predefine the number of clusters for the clustering of network data based on the similarity of link information; FCM is used as the baseline of performance and we implemented it based on the similarity of content information.

4.5.1 Evaluation Metrics

To assess the degree of matching between the resultant clusters and ground truth, we adopted two evaluation metrics and they were *NMI* [83] and *Accuracy*. These two metrics are widely used to measure how well the clustering result matches the ground truth.

The purpose of *NMI* is to information-theoretically indicate the degree of matching the ground truth. Assuming that $\mathrm{T} = \{\mathrm{T}_f\} (1 \leq f \leq k)$ is the ground truth of clusters of vertices in V, *NMI* is defined as

$$NMI = \frac{\sum_{f_1=1}^{k}\sum_{f_2}^{k} n_{\mathrm{C}_{f_1},\mathrm{T}_{f_2}} \log\left(\frac{n_{\mathrm{v}} n_{\mathrm{C}_{f_1},\mathrm{T}_{f_2}}}{n_{\mathrm{C}_{f_1}} n_{\mathrm{T}_{f_2}}}\right)}{\sqrt{\left(\sum_{f=1}^{k} n_{\mathrm{C}_f} \log\frac{n_{\mathrm{C}_f}}{n_{\mathrm{v}}}\right)\left(\sum_{f=1}^{k} n_{\mathrm{T}_f} \log\frac{n_{\mathrm{T}_f}}{n_{\mathrm{v}}}\right)}} \tag{30}$$

where n_{C_f} is the number of vertices in C_f, n_{T_f} is the number of vertices in T_f, and $n_{\mathrm{C}_{f_1},\mathrm{T}_{f_2}}$ is the number of common vertices in both C_{f_1} and T_{f_2}.

For the measure *Accuracy*, the mapping function $Z:\mathrm{C}_{f_1} \to \mathrm{T}_{f_2}$ needs to be determined before we compute the value of *Accuracy*. To decide Z, we first obtain the results of $n_{\mathrm{C}_{f_1},\mathrm{T}_{f_2}}$ for all combinations of C_{f_1} and T_{f_2} in C and T respectively. For each iteration, starting from the largest $n_{\mathrm{C}_{f_1},\mathrm{T}_{f_2}}$, we select T_{f_2} as the corresponding cluster of C_{f_1} in T, then add the mapping $\mathrm{C}_{f_1} \to \mathrm{T}_{f_2}$ to Z, and disregard both C_{f_1} and T_{f_2} in the following iterations. The iteration process will end till each T_{f_2} in T has a corresponding unique cluster C_{f_1} in C. After obtaining the mapping function Z, the value of *Accuracy* is given below.

$$Accuracy = \frac{1}{n_{\mathrm{v}}} \sum_{f=1}^{k} n_{\mathrm{C}_f, Z(\mathrm{C}_f)} \tag{31}$$

According to the definitions of *NMI* and *Accuracy*, the values of *NMI* and *Accuracy* are larger if the clustering result $\{\mathrm{C}_f\}$ is better in terms of the degree of matching with T. That is to say, if $\{\mathrm{C}_f\}$ completely matches T, both *NMI* and *Accuracy* will be 1, which is the maximum value they can take.

4.5.2 Experiment Setup

There were three experiments we have setup for performance evaluation. The first experiment was designed to assess the ability of identifying clusters that match our expectations. For the second experiment, we focused on studying the respective influence of the four parameters. The last experiment was to estimate the scalability of FC-AG. Hence, synthetic networks were created with different sizes in the last experiment and the largest synthetic network was composed of hundreds of thousands of vertices and millions of edges.

For each of synthetic networks used in the experiments, we created four clusters, each of which was composed of vertices that were densely connected and that were associated with similar attributes. To create such a synthetic network, the respective probabilities of intra-connection and inter-connection were used to achieve the purpose that the links among vertices in the same clusters were much more than those among vertices in the different clusters. For the content information, sixteen to twenty attributes were generated to compose Λ for each synthetic network. Attributes of Λ were associated with vertices in the sense that vertices in the same cluster were similar in several attributes while vertices from different clusters shared few attributes.

All approaches have run 30 trials to avoid the potential bias resulted from random initializations and the average results of *NMI* and *Accuracy* were used for performance comparison. Regarding the parameter setting for each approach, we followed the recommendations provided in the related papers to determine the proper values of parameters involved; if no recommendation was provided, the combination of values with the best performance was selected for comparison experiments with other approaches after trying different combinations of values.

Regarding the implementation of the proposed approach, we implemented it with JAVA. For FCM, we used the APIs provided by Mahout [61] for implementation. The

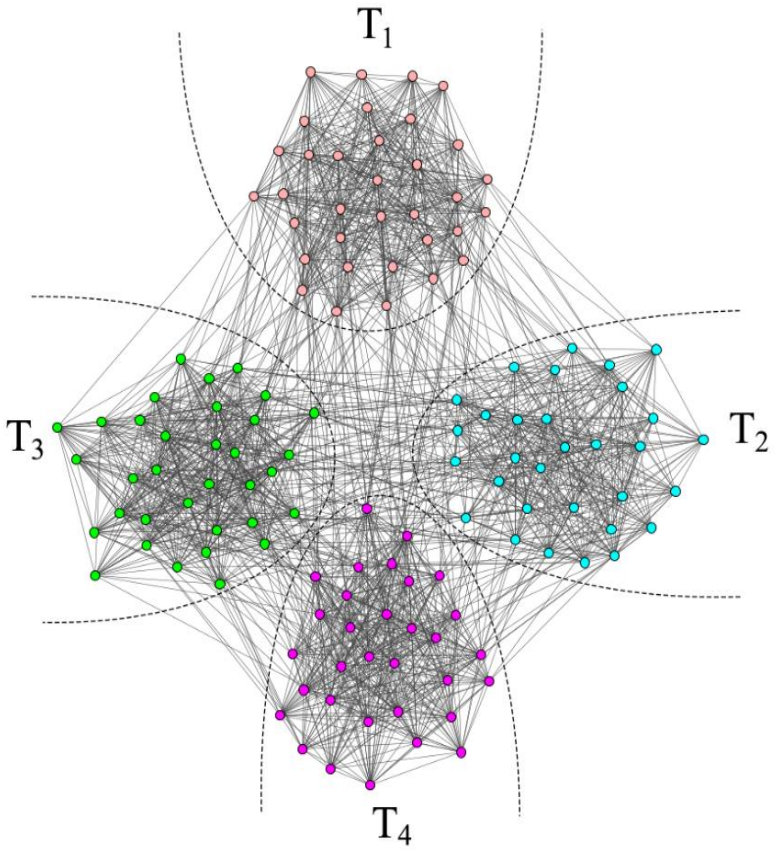

Figure 5. The synthetic network used in the experiment of assessing the resultant clusters. Four clusters are generated and they are highlighted with different colors.

source codes of the other approaches were shared by their authors. The experiments were performed on a machine with Intel Core i5-2310 processors and 8GB RAM.

4.5.3 Experiment Results

4.5.3.1 Assessing the resultant clusters

To assess the ability of FC-AG in terms of identifying clusters matching our expectations, we created a small synthetic network that was composed of 131 vertices and 1782 links. The graph of this synthetic network is presented in Figure 5, where vertices in the same cluster are highlighted with the same color and vertices in different clusters are separated by dashed lines.

From Figure 5, we observed that most of vertices in the same cluster were densely connected with each other, but boundary vertices, which were located near dashed lines, also had comparable links with vertices from different clusters. That is to say, purely replying on topology information may fail to group these boundary vertices in the correct clusters. Hence, the additional consideration of attribute information can overcome this problem.

The experiment results for the synthetic network in Figure 5 were given in Table 5. Overall, FC-AG outperformed the other approaches in terms of both *NMI* and *Accuracy*. Since the scores of *NMI* and *Accuracy* were almost equal to 1, the clusters identified by FC-AG were very close to the ground truth. When compared with the other approaches, FC-AG was better by 8%, 20%, 185% and 259% than SGC, CESNA, FCM and iTopicModel respectively in terms of *NMI* and was better by 19%, 41%, 102% and 111% than SGC, CESNA, iTopicModel and FCM respectively in terms of *Accuracy*. Hence, the ability of FC-AG has been verified as the resultant clusters satisfy the aforementioned requirements.

Table 5. The Scores of *NMI* and *Accuracy* for the synthetic network in Figure 5

	NMI	***Accuracy***
FC-AG	$0.97^{(1st)}$	$0.99^{(1st)}$
iTopicModel	0.27	0.49
CESNA	$0.81^{(3rd)}$	$0.7^{(3rd)}$
SGC	$0.9^{(2nd)}$	$0.83^{(2nd)}$
FCM	0.34	0.47

Although FC-AG, iTopicModel and CESNA are proposed to perform the clustering task by considering both topology and attribute information, their performances on the synthetic network were quite contrasting. In particular, FC-AG obtained the best scores of *NMI* and *Accuracy* while iTopicModel were the worst one. The main reason for the unsatisfactory performance of iTopicModel is due to the fact of ignoring the dense link structures among vertices in the same cluster, as iTopicModel only made

use of link information to generate the Random Markov Fields for vertices. In contrast, as CESNA also considered the dense link structures as the one of properties of clusters identified, it obtained a better performance in the synthetic network.

The remaining two approaches, i.e., SGC and FCM, utilized the link information and the content information respectively to perform the clustering task. Based on their performances, we found that the link information was more useful for the clustering on the synthetic network. The reason why SGC performed worse than FC-AG is that FC-AG also considered the attribute information when identifying clusters so that the boundary vertices that were misclassified by SGC were correctly classified by FC-AG.

In sum, the promising performance of FC-AG verified the rationality of the maximization optimization problem we formulate for the clustering of network data and also show the efficiency of FC-AG.

4.5.3.2 Parameter Sensitivity Analysis

As a part of the maximization optimization problem (13), the four parameters ∂, β, ϕ and θ are introduced to adjust the optimization process. Although we analyze the problem of how to choose proper values for these four parameters in 4.4.4, their respective influences on the performance of FC-AG are yet to be investigated. In this section, we will take the synthetic network in Figure 5 as an example again and analyze the influences of these four parameters on the performance of FC-AG.

Before presenting the analysis, we first explain how we selected proper values of ∂, β, ϕ and θ for the synthetic network in Figure 5. According to (28) and (29), we obtained the values of a_{avg} and d_{avg} as -0.17 and 0.21 respectively. Hence, following the suggestion that $\partial | a_{avg} |$ and βd_{avg} should have the same order of magnitude, we set both the values of ∂ and β equal to 0.5. For the values of θ and ϕ, we adopted the recommendation and set the values of them as 0.5 too. When evaluating their

respective influences on the performance of FC-AG, we adopted the strategy of alternatively changing the parameters. For example, if we would like to study the influence of ∂, we will try to change the value of ∂ while fixing the values of the other parameters.

In Figure 6, four subfigures, from top to down, respectively show the impacts of ∂, β, ϕ and θ on the performance of FC-AG in terms of *Accuracy* and *NMI*. Regarding sensitivity results of ∂, the scores of *Accuracy* and *NMI* had a steadily increase when the value of ∂ changed from 0 to 0.5. In particular, the scores of *Accuracy* and *NMI* raised by 65% and 33% respectively at $\partial = 0.5$ when compared with $\partial = 0$. However, the upward trend ended at $\partial = 0.5$ and the performance of FC-AG become stable when $\partial > 0.5$ as indicated by the small change in the scores of *Accuracy* and *NMI*. For the network in Figure 5, since the value of ∂ indicates to what extent the content information has been considered by FC-AG, the increase in the amount of content information resulted in a considerable improvement for both *Accuracy* and *NMI* when $\partial \leq 0.5$. At $\partial = 0.5$, the scores of *Accuracy* and *NMI* were all close to their maximum values and hence the performance of FC-AG could not be improved further even when more content information has been considered.

For the parameter β, similar to the meaning of ∂, the value of β indicates to what extent the link information has been considered by FC-AG. When looking at the results of the sensitivity test of β, we found that the fluctuation ranges of *Accuracy* and *NMI* were much smaller than those resulted from the change of ∂. In particular, the change of β within the range [0, 0.5] did not affect much on the performance of FC-AG, but both the curves of *Accuracy* and *NMI* dropped at different degrees when $\beta > 0.5$. After investigating the results, we noted that the peripheral vertices that also had many links with vertices in different clusters become more difficult to be classified when the amount of link information increased, thus making a negative impact to the performance of FC-AG.

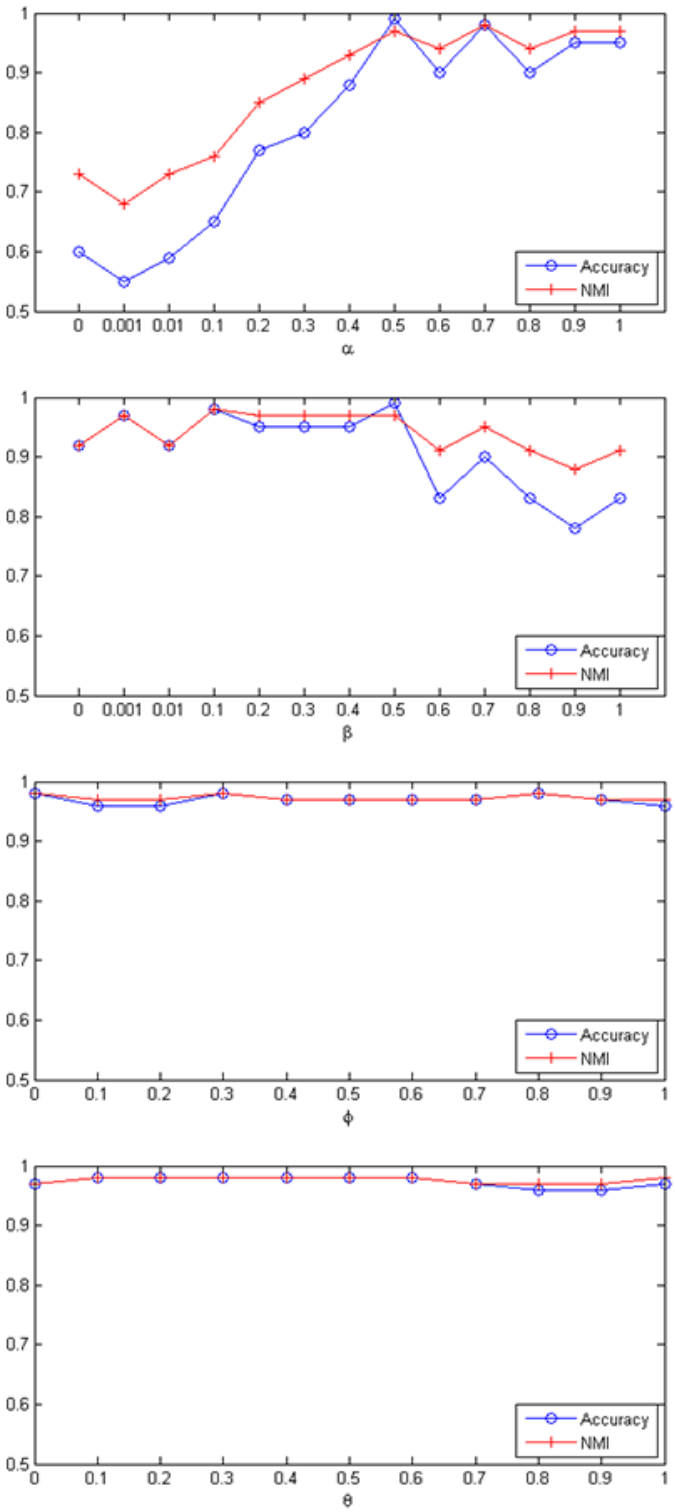

Figure 6. Results of parameter sensitivity tests on the performance of FC-AG.

Comparing the impacts of ∂ and β, we noted that the performance of FC-AG was more sensible to the change of ∂. Hence, a conclusion could be reached that the

contribution made by the content information was more significant than that from the link information for the clustering of the network in Figure 5.

For both ϕ and θ, their impacts on the performance of FC-AG were limited, as the scores of *Accuracy* and *NMI* did not change much when their values increased from 0 to 1. It should be noted that θ is associated with third property of resultant clusters according to (15) and (16). But for the network in Figure 5, only considering the first two properties can guarantee the promising performance of FC-AG and hence the impact of θ was limited.

In sum, for the synthetic network in Figure 5, we found that the performance of FC-AG was more sensitive to ∂ when compared with the other three parameters. For the impact of β, since FC-AG adopts a feature selection to preprocess the content information so that the relevant information can be retained, the contribution made by the link information is not as significant as purely resting on the link information, especially when the content information is highly related to the ground truth clusters in network data, such as the network in Figure 5.

4.5.3.3 Scalability

To evaluate the salability of FC-AG, we performed the experiment to measure the running time on synthetic networks by increasing the size in terms of the number of vertices. To generate these synthetic networks, we set the probabilities of intra-connection and inter-connection as 0.1 and 0.01 respectively. For the content information, we adopted the same procedure as we did to create the synthetic network in Figure 5. The number of clusters to be identified was four for all synthetic networks. The size of synthetic networks varied from hundreds of vertices to hundreds of thousands of vertices as indicated by the x-axis in Figure 7.

For each synthetic network, rather than comparing all approaches in Table 5, we run the top three approaches, i.e., FC-AG, SGC and CESNA, for 30 trials and used the

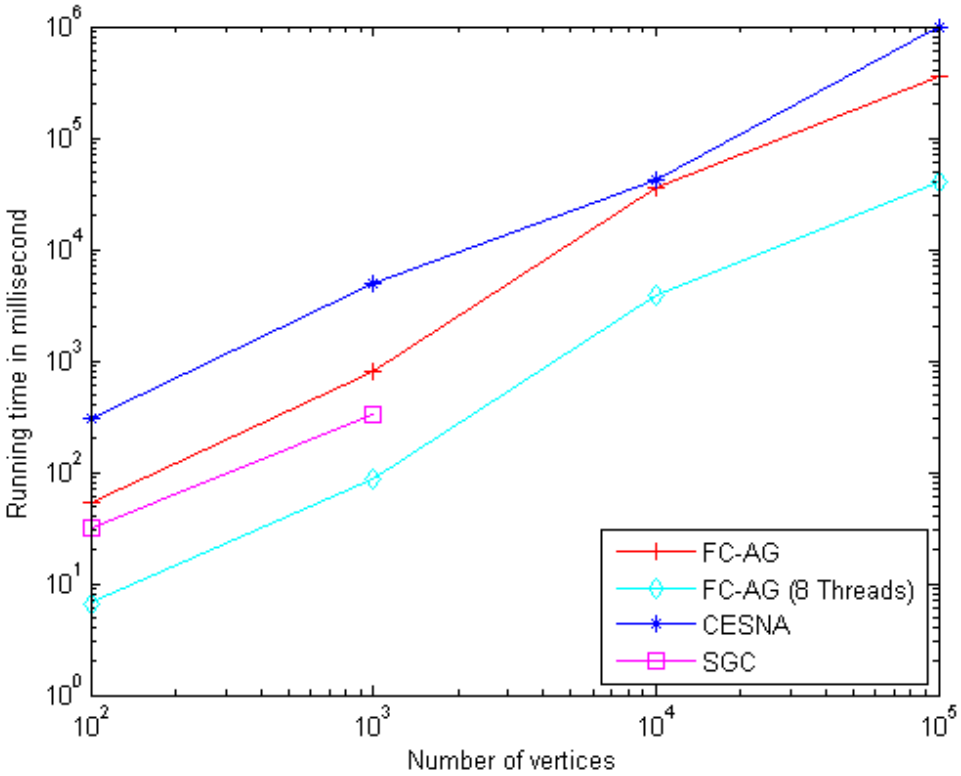

Figure 7. The experiment results of assessing scalability for FC-AG, CESNA and SGC.

average time to draw the respective scalability curves. The experiment results of scalability are presented in Figure 7. It is seen that both FC-AG and CESNA had an almost linear runtime in the network size. In particular, FC-AG obtained a better performance than CESNA, as the scalability curve of FC-AG was always below the one of CESNA in Figure 7. In addition to the promising performance of FC-AG, we also show the scalability of parallelized FC-AG. Obviously the parallelization of FC-AG considerably improved the efficiency of FC-AG. Regarding SGC, its performance was the best for small networks due to the fact that SGC does not consider the content information during clustering. We did not obtain the running time of SGC on large networks, as the configuration of our machine could not fulfill the computational capacity required by SGC for large networks.

Overall speaking, FC-AG has an almost linear runtime in the network size for the concern of scalability. Besides, FC-AG is more efficient than the state-of-the-art clustering approach, i.e., CESNA.

4.6 Applications with Real Data

In addition to the experiments on synthetic network data, we also applied FC-AG to two practical applications with real network data so that the advantages of FC-AG can also be demonstrated when used to solve practical clustering problem. The two applications were document classification and the identification of social communities. The experiment results will be given in the rest of this section.

4.6.1 Document Classification

4.6.1.1 Experiment setup

To evaluate the performance of FC-AG for document classification, we conducted experiments with two benchmark datasets, Cora Dataset and Citeseer dataset. Generally speaking, the two benchmark datasets are citation networks and they are published by [78]. Both of these two benchmark datasets are available to download[2].

Cora Dataset is a network composed of 2708 scientific publications, which are classified into one of seven classes, including Case-based Reasoning, Genetic Algorithms, Neural Networks, Probabilistic Methods, Reinforcement Learning, Theory and Rule Learning. The Cora Dataset has 5429 links. The keywords of each publication are from a dictionary of 1433 unique words. Each publication is then described by a 0/1-valued word vector indicating the absence/presence of the corresponding word from the dictionary. The number of clusters is set to be k =7, which is equal to the number of classes in Cora Dataset.

[2] http://www.cs.umd.edu/projects/linqs/projects/lbc/index.html

Citeseer Dataset is composed of 3312 scientific publications and 4732 links among them. Each publication in Citeseer Dataset is classified into one of six classes, including Agents, Artificial Intelligence, Database, Information Retrieval, Machine Learning and Human-Computer Interaction. The keywords of each publication are from a dictionary of 3703 unique words. Each publication is then described by a 0/1-valued word vector indicating the absence/presence of the corresponding word from the dictionary. The number of clusters is set to be k =6, which is equal to the number of classes in Citeseer dataset.

For performance evaluation, in addition to the clustering approaches used in Section 4, we also compared with several state-of-the-art approaches specifically proposed for document classification and they were FC-MR [58] and Spectral CoClustering (SCC) [25]. FC-MR is a coclustering approach that makes use of the homogeneity of the dependency between documents and keyword information to perform its tasks. SCC models the document collection as a bipartite graph between documents and words and then tackle the problem of document classification using bipartite graph partitioning.

4.6.1.2 Experiment Results

The experiment results of Cora and Citeseer datasets are reported in Table 6 and Table 7 respectively.

Table 6. The Scores of *NMI* and *Accuracy* for Cora Dataset

	NMI	***Accuracy***
FC-AG	$0.43^{(2nd)}$	$0.53^{(2nd)}$
iTopicModel	0.18	0.37
CESNA	$0.47^{(1st)}$	$0.59^{(1st)}$
SGC	0.01	0.3
FCM	0.07	0.26

FC-MR	0.19(3rd)	0.41(3rd)
SCC	0.11	0.29

Table 7. The Scores of *NMI* and *Accuracy* for Citeseer Dataset

	NMI	***Accuracy***
FC-AG	0.4(1st)	0.65(1st)
iTopicModel	0.22(3rd)	0.44(3rd)
CESNA	0.16	0.42
SGC	0.03	0.21
FCM	0.13	0.33
FC-MR	0.25(2nd)	0.51(2nd)
SCC	0.18	0.35

From Table 6 and

Table 7, it is seen that FC-AG had a promising performance for document classification as it was the second approach for the Cora dataset and was the best approach for the Citeseer dataset in terms of *NMI* and *Accuracy*.

In particular, for the Cora dataset, according to Table 6, FC-AG performed better by 29% than the third best approach (i.e., FC-MR) in terms of *Accuracy*, and the advantage of FC-AG become larger for the *Accuracy* metric where FC-AG was 1.26 times more than FC-MR. However, FC-AG did not perform better than CESNA, as it was worse by 9% and 11% than CESNA for *NMI* and *Accuracy* respectively. To find out the reason why CESNA outperformed FC-AG, we investigated the clustering results of CESNA and found that not all documents were classified by CESNA at each trial. The unclassified rate of documents was 37% on average. That is to say, the superior performance of CESNA for the Cora dataset was resulted from the fact that CESNA filtered out documents that were difficult to be classified and only performed clustering on documents that fulfilled the requirements of CESNA. Since FC-AG

performed the clustering on all documents involved in the Cora dataset, FC-AG was better than CESNA in this regard.

When looking at the results of Citeseer dataset in Table 7, we found that FC-AG obtained the best performance among all approaches. In particular, for *NMI*, FC-AG was better by 60% and 82% than the second best approach, i.e., FC-MR and the third best approach, i.e., iTopicModel, respectively; for *Accuracy*, FC-AG was better by 27% and 48% than FC-MR and iTopicModel respectively. Regarding the performance of CESNA on the Citeseer dataset, it did not perform as well as it obtained for the Cora dataset and furthermore the unclassified rate, as high as 77%, was more than double that of the Cora dataset. It is for this reason that CESNA obtained an unsatisfactory performance for *NMI*. Concerning the major difference between the Cora and Citeseer datasets, we noted that the Citeseer dataset was much sparser than the Cora dataset. Hence, a conclusion could be reached that the performance of CESNA is sensitive to the sparsity of network data, whereas the performance of FC-AG is not affected much.

Comparing with the performance of each approach in 4.5, we found that both FC-AG and CESNA performed steadily as they always obtained a promising performance in both synthetic and real network data. However, not all approaches that obtained a promising performance on synthetic network data could also have a good performance for practical applications. Taking SGC as an example, SGC was the second best approach for synthetic datasets but its performance dropped considerably in the experiment of document classification. Hence, considering the complexity of network data in practical applications, resting on only a single source of information is insufficient and risky for the task of clustering and that is also the motivation for the proposal of FC-AG, with which we can exploit the fully potential knowledge in AGs.

Overall, we noted that across all datasets and evaluation metrics, FC-AG yielded a promising performance in all trials. Comparing FC-AG to approaches that also additionally considered the content information for clustering (i.e., iTopicModel and CESNA), we noticed that FC-AG achieved better performance, as it filtered out

irrelevant content information by introducing a content relevance measure. Similarly, FC-AG outperformed SGC, which only focused on the similarity of link structures. FC-AG also obtained a better performance than FCM, which only concentrated on the similarity of content information. Furthermore, FC-AG never performed worse than state-of-the-art approaches proposed for document clustering, i.e., FC-MR and SCC. Thus, if we would like to classify all documents while obtaining a satisfactory performance, FC-AG is preferred.

4.6.2 Social Community Detection

4.6.2.1 Experiment Setup

The task of identifying social communities has been being popular recently due to the rapid development of social networks and it plays an important role to help researchers to understand and exploit these networks efficiently [33]. Hence, we applied FC-AG to identify communities from ego-networks extracted from Facebook and Twitter and then evaluated its performance by comparing with iTopicModel, CESNA, SGC and FCM. Both of Facebook Dataset and Twitter Dataset are available to download from the Stanford Large Network Dataset Collection[3].

Facebook Dataset is composed of 4089 vertices and 170714 edges. The ground truth communities are defined by social circles that are manually labeled by the owner of the corresponding ego-network and the number of communities, or the value of *k*, is 193. For the content information, user profiles, such as gender, job titles and institutions, are used as the attributes of vertices. The number of attributes that a vertex can possess is 175.

Twitter Dataset is much larger than Facebook Dataset, as there are 125120 vertices and 2248406 edges in Twitter Dataset. The number of ground truth communities is

[3] http://snap.stanford.edu/data/index.html

3140 and the number of attributes that are available to be assigned to vertices is 33569. Unlike Facebook Dataset whose attributes are user profiles, the attributes used in Twitter Dataset are extracted from the hashtags used by users in their tweets.

4.6.2.1 Experiment Results

The experiment results are given in Table 8 and Table 9 for Facebook Dataset and Twitter Dataset respectively.

Table 8. The Scores of *NMI* and *Accuracy* for Facebook Dataset

	NMI	***Accuracy***
FC-AG	$0.49^{(2nd)}$	$0.51^{(1st)}$
iTopicModel	0.34	0.32
CESNA	$0.58^{(1sd)}$	$0.46^{(2nd)}$
SGC	$0.39^{(3rd)}$	$0.39^{(3rd)}$
FCM	0.14	0.34

Table 9. The Scores of *NMI* and *Accuracy* for Twitter Dataset

	NMI	***Accuracy***
FC-AG	$0.38^{(1st)}$	$0.68^{(1st)}$
iTopicModel	0.24	0.53
CESNA	$0.31^{(3rd)}$	$0.65^{(3rd)}$
SGC	$0.37^{(2nd)}$	$0.66^{(2nd)}$
FCM	0.22	0.58

For Facebook Dataset, FC-AG was the second best approach in terms of *NMI*, as its *NMI* score was lower by 15.5% than that of CESNA. However, since the unclassified rate of CESNA in Facebook Dataset was as much as 18%, CESNA took advantage of the high unclassified rate to obtain a superior performance of *NMI*. Even though, the performance of CESNA in terms of *Accuracy* was not as well as that of FC-AG, which performed better by 11% than CESNA. In contrast to the performance for the

application of document classification, SGC was the third best approach for identifying social communities from Facebook dataset.

In the experiment of Twitter Dataset, FC-AG obtained the best performance in terms of both *NMI* and *Accuracy*. However, the advantage of FC-AG was not as significant as it performed on the datasets of Cora, Citeseer and Facebook, as the difference in both *NMI* and *Accuracy* between FC-AG and the second best approach, i.e., SGC, was much smaller. In this case, considering the contrasting performance of SGC in the applications of document classification and social community detection, we believe that the homogeneity in the link information was more easily to be observed in social communities. But for cases where we do not have prior knowledge to learn the link structures before clustering, FC-AG is preferred.

In sum, we believe that the promising performance of FC-AG can be an indication that FC-AG can identify clusters more accurately by combining content relevance and link structure.

4.7 Case Studies

In this section, we present an in-depth analysis to a cluster identified by FC-AG from Twitter Dataset so that the benefits of FC-AG can be demonstrated.

The topological structure of the selected cluster is described in Figure 8. It is observed that the part in the solid circle is much denser than the peripheral part of this cluster that is outside the circle. In particular, for vertices in the solid circle the average degree is 11.1, but for vertices outside the circle the average degree is only 5.4. Hence, considering the approaches that make use of link structures, it is difficult for them to identify the peripheral vertices in Figure 8 as they are not connected as dense as those in the circle. In this regard, there is a necessity for us to leverage the content information so that such kind of clusters could be identified.

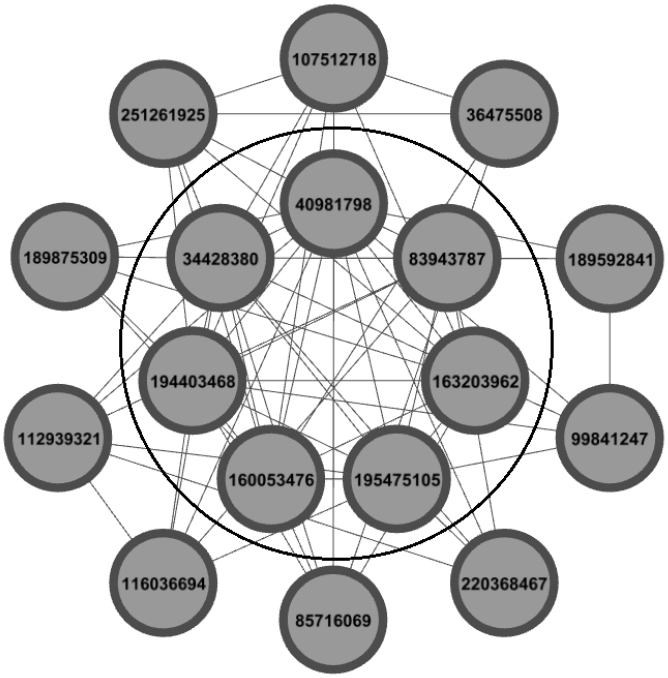

Figure 8. The topological structure of a ground truth cluster in Twitter Dataset. FC-AG completely identified this cluster while the part inside the solid circle was the cluster identified by CESNA.

Due to the page limit, we cannot list the attributes of all vertices in Figure 8. Thus, we selected two representative vertices 189875309 and 34428380 and show their attributes in Figure 9. From Figure 9, we noted that 189875309 and 34428380 did not have any attribute in common. That is to say, neither model-based approaches nor distance-based approaches will classify them into the same cluster because of their dissimilar attributes and different attribute distributions. We further noted that not only 189875309 but also other vertices outside the circle shared few attributes with their adjacent vertices located inside the circle. It is for this reason that CESNA can only detected the part in the solid circle.

FC-AG overcome this problem with the use of content relevance. To better illustrate it, we present the associated attribute values that were related to the discovery of this cluster in Table 10. There were three pairs of associated values identified by FC-AG.

Figure 9. The attributes of two twitter users 189875309 and 34428380 in Twitter Dataset.

To verify whether these associated values were meaningful, we checked these hashtags in the Twitter website. Taking Paramorelovers and absolutepunk as an example, the hashtag absolutepunk was referred to an online community that mainly focuses on artists working in punk music, and Paramorelovers represented the fans of Paramore which is a punk band. In this regard, the association between absolutepunk and Paramorelovers can be verified and hence it is reasonable to reckon them as the relevant content information that is useful for the purpose of clustering. Based on a fuzzy-based clustering framework, FC-AG successfully identified the cluster in Figure 8 by combing the relevant content information and the dense link structure.

Table 10. Associated Attribute Values That Were Related to the Discovery of The Cluster in Figure 8

Attribute Value	**Attribute Value**	**Strength**
Paramorelovers	absolutepunk	0.92
mthardcore	absolutepunk	0.85
absolutepunk	LGSTV	0.84

4.8 Conclusion

In this paper, we propose FC-AG to perform the clustering task for AGs by considering content association and link structure simultaneously. To accurately identify clusters from the perspective of attribute information, we make use of associative patterns to weight the association between pairwise vertices based on their attributes so that the vertices in the same cluster are expected to be highly associated. For link structure, we follow the intuitive property that clusters shall be composed of vertices that are densely connected. Then a maximization optimization problem is formulated under a fuzzy-based clustering framework. For each of vertices, the fuzziness of memberships can be estimated using content association and link structure. Therefore, FC-AG can identify clusters that fulfill the requirements aforementioned. To evaluate the performance of FC-AG, we have performed extensive experiments on both synthetic and real networks. The experiment results show that FC-AG significantly improves the clustering quality over the state-of-the-art approaches in terms of accuracy and efficiency and hence indicate that FC-AG has a promising performance for the clustering of network data by considering the attribute and topology information simultaneously. Furthermore, FC-AG has a linear runtime when the size of network data increases.

Chapter 5 Unsupervised AG Clustering

5.1 Overview

In the last chapter, we have studied the problem of AG clustering by proposing FC-AG. However, since FC-AG tackles the problem based on fuzzy clustering, the number of clusters has to be determined in advance. Recognizing that there are also many applications where the number of clusters is unknown, such as the identification of protein complexes from PPINs, we fell that there is a necessity for us to develop a clustering approach in an unsupervised manner so that predetermining the number of clusters can be avoided. Hence, in this chapter, we introduce two unsupervised AG clustering approaches MCL-AG and CAP-AG from different perspectives. In particular, MCL-AG performs the task through a markov clustering process based on associative patterns while CAP-AG identified clusters from AGs by considering the attribute preferences and link structure simultaneously.

Given an AG, MCL-AG identifies clusters by first constructing a weighted AG (wAG) based on the DOA measure. Even though higher DOA values may not necessarily mean the significance of edges, it does mean that the vertices involved may be in the same cluster as their attributes are more significantly associated. Given a wAG, MCL-AG therefore makes use of markov clustering process to find subgraphs that are relatively denser on one hand and contains vertices that have significant associations with each other on the other. Once the subgraphs are discovered, a partitioning process is then performed by MCL-AG for each subgraph discovered to find partitions that have, according to the DOA measure, closely associated vertices. These partitions identified correspond to the set of clusters that can be identified in an AG.

In addition to MCL-AG, we also propose another unsupervised clustering approach, namely CAP-AG. Regarding the use of topology information, both of MCL-AG and

CAP-AG follow the same intuitive property that vertices in the same cluster are densely connected. But for the attribute information, in contrast to MCL-AG that considers the attribute information as a whole according to the DOA measure, CAP-AG explores the feasibility of measuring the likelihood of pairwise vertices being grouped in the same cluster based on a subset of attributes instead of all attributes so that clusters can be identified more accurately and flexibly. To do so, CAP-AG employs a likelihood matrix to represent to what extent pairwise vertices are likely to be identified in the same cluster. If the likelihood between two vertices is relatively high, it means that they are more likely to be identified in the same cluster. For each vertex, CAP-AG also assigns a corresponding preference vector to quantitatively indicate the contribution of each attribute during the clustering process. Then the problem of identifying clusters from AGs is formulated into a maximization optimization problem. Since there are two variables, i.e., the likelihood matrix and the preference matrix, to be optimized, CAP-AG adopts the strategy of alternatively optimizing the likelihood matrix and the preference matrix through an iterative procedure to tackle the optimization problem. This procedure initially starts from a random guess of both the likelihood matrix and the preference matrix of all vertices and then iteratively improves the quality of the clustering until convergence.

To demonstrate the respective advantages of MCL-AG and CAP-AG, we apply them to the application of identifying protein complexes from PPINs and also compare their performances with existing state-of-the-art approaches proposed for the identification of protein complexes. The experiment results show that both of these two approaches outperform most of state-of-the-art approaches.

The rest of this paper is organized as follows. The details of MCL-AG and CAP-AG are presented in Section 5.2 and Section 5.3 respectively. We then apply MCL-AG and CAP-AG to the application of identifying protein complexes from PPINs and analyze the results in Section 5.4. Finally, we end the paper with a conclusion in Section 0.

5.2 MCL-AG

Given an AG, MCL-AG performs its tasks in several steps:

1) **Constructing wAG**. The DOA measure is computed for each edge in the AG and then assigned as a weight to the corresponding edge. With these weights, a wAG is obtained.

2) **Discovering Subgraphs in wAG**. MCL-AG makes use of a markov clustering algorithm, i.e., MCL [27], to discover a set of dense subgraphs (i.e., S) from the wAG.

3) **Identifying Clusters**. According to how closely associated each vertex is to each other in a partition, subgraphs are further partitioned into clusters composed of closely associated vertices.

These four steps are given in details in the followings.

5.2.1 Constructing wAG

Given two adjacent vertices v_i and v_j ($e_{ij} \in \mathrm{E}$), the association between them can be computed by the DOA measure introduced in 2.4.2 and we take this value as the weight of e_{ij}. Once we obtain the weights of all edges in AG, a wAG can be constructed.

5.2.2 Discovering Subgraphs in wAG

The purpose of this step is to discover a set of dense subgraphs, i.e., $S = \{s_i\}$, from the wAG obtained in the last step using a markov clustering algorithm. For the current version of MCL-AG, the MCL is used for this purpose. The reason why we choose to

use MCL is that MCL has been known to perform relatively well and is very robust in general. Compared with the other algorithms, MCL can also be more easily modified to take into consideration the attribute information that we are interested in. For MCL, the attribute information can be integrated into the adjacency matrix of a network to consider so that such information can be used to control the search paths through the random walks.

To use MCL, MCL-AG represents the input graph in a transition matrix and makes use of it to implement the flow paths. Such a transition matrix can not only provide information as to whether a vertex represented by a particular row is connected with another represented by a particular column, it can also reveal the probability of flow moving from a row vertex to a column vertex. The probability can simply be computed based on the degree of a vertex as shown in the transition matrix of a graph.

For the case of an unweighted graph, for example, if the degree of a vertex is 2, the probability for each path that goes out from this vertex is 0.5. However, for a weighted graph such as the wAG, the probability of the direction of flow movement has to be determined, not just by the degree of vertex, but also by the weights of the edges concerned. For example, assuming that one of the edges in our last example that flows out from the vertex has the weight of 0.2 and the other edge has the weight of 0.3, then the flow probabilities should be changed, after normalization, to 0.4 (i.e., 0.2/(0.2+0.3)=0.4) and 0.6 respectively (i.e., 0.3/(0.2+0.3)=0.6).

Since the transition matrix we have already contains the weights of the edges, MCL-AG simply performs a random walk to simulate the flow through matrix multiplication so that neighbors can be validated if it is possible for them to appear in the same cluster. After expansion as flowing from the previous matrix and through the original transition matrix, MCL uses the inflation operator to regulate the granularity of clusters to obtain the effect that flows are much easier for dense regions. By iteratively executing the operators of expansion and inflation of MCL, MCL-AG explicitly discovers a set of subgraphs, S .

With the flow movement considered in the construction of graph clusters, MCL-AG is therefore able to take into consideration both attribute value information directly and indirectly as such information is used to determine the weights which MCL-AG uses in the discovering of the set of subgraphs S. The subgraphs in S can therefore be formed by achieving a balance between cluster density and attribute value associations.

5.2.3 Identifying Clusters

Each subgraph discovered in the previous step is examined in this final step to see if it can be partitioned according to how much the vertices in the cluster are associated with each other. To identify such partitions, we define a range, $[\mu w_{max}, w_{max}]$, where w_{max} is the maximum weight of an edge in a cluster in S and μ, $0 \leq \mu \leq 1$ is used to control how close the vertices in a cluster associate with each other.

Hence, in this final step, a subgraph in S is partitioned into smaller subgraphs, which MCL-AG considers as a cluster, so that each such cluster contains edges that have weights that are relatively close to each other. To partition a subgraph, MCL-AG adopts an iterative process. First, it selects the edge with the maximum weight (i.e., w_{max}) and based on it, forms a cluster, r. A depth-first search (DFS) strategy is then used to find edges among the adjacent edges of this edge that have weights that are within the range $[\mu w_{max}, w_{max}]$. Once an edge is found, it is added to the cluster, r, that is being formed. Edges that are not within the pre-defined range are removed from the given subgraph so as to prepare for the next iteration. Hence, by looping the partitioning operation, we can identify a set of clusters from S. Figure 10 presents a detailed description of identifying clusters.

Algorithm 1: Identifying Clusters

Input: S, μ

Output: R as a set of clusters identified

for all s in S **do**
  E_s is the set of edges in s
  while $E_s \neq \phi$ **do**
    create a new cluster r
    the set of examined edges $E_{exa} \neq \phi$
    $e_{max} \in E_s$ is the edge with the maximum weight w_{max}
    $min = \mu w_{max}$
    CALL $DFS(s, r, e_{max}, min, E_{exa})$
    $R = R \bigcup r$
    remove all edges in r from E_s
  end while
end for
return R

Procedure $DFS(s, r, e, min, E_{exa})$

$E_{adj} \subset E_s$ is the set of adjacent edges of e
$E_{exa} = E_{exa} \bigcup e$
$r = r \bigcup$ the set of vertices connected by e
for all $e_{adj} \in E_{adj}$ **and** $e_{adj} \notin E_{exa}$ **do**
  if $w_{adj} \geq min$ **then**
    CALL $DFS(s, r, e_{adj}, min, E_{exa})$
  else
    $E_{exa} = E_{exa} \bigcup e_{adj}$
  end if
end for

Figure 10. The pseudo code of identifying clusters.

5.3 CAP-AG

5.3.1 Problem Statement

According to Section 2.3, an AG is represented as a 3-element tuple $\mathrm{AG} = \{\mathrm{V}, \mathrm{E}, \Lambda\}$. For convenience, we use a $n_{\mathrm{V}} \times n_{\mathrm{V}}$ matrix $\mathbf{T} = [t_{ij}]$ to represent the topology of AG and t_{ij} is determined according to (32).

$$t_{ij} = \begin{cases} 1, & e_{ij} \in \mathrm{E} \\ 0, & \text{otherwise} \end{cases} \tag{32}$$

The problem of AG clustering we are working on is to identify a set of clusters, i.e., C, from AG. In each cluster of C, vertices are densely connected and they are at least similar in some attributes. We use the likelihood matrix $\mathbf{W} = [w_{ij}]$ $(1 \le i, j \le n_{\mathrm{V}})$ to represent the likelihoods of being grouped in the same cluster for all pairwise vertices in AG. For example, $w_{ij} \in [0,1]$ denotes how likely v_i and v_j are being identified in the same cluster. The larger the value of w_{ij} is, the more likely v_i and v_j are grouped in the same cluster. Therefore, for e_{ij}, we can partition the AG by removing it if w_{ij} is less than a predefined threshold, i.e., w_{min}. After disregarding all this kind of edges, we can obtain C.

Regarding the similarity of attribute information, $\mathbf{A} = \{\mathbf{A}_m\} (1 \le \mathrm{m} \le n_{\Lambda})$ is a set of similarity matrices obtained by comparing the attributes of vertices. Specifically speaking, $\mathbf{A}_m = [a_{ij}^m]$ is the similarity matrix based on the attribute information of Λ_m and a_{ij}^m denotes the similarity score of v_i and v_j in terms of Λ_m. In this work, we define a_{ij}^m as below.

$$a_{ij}^{m}=\begin{cases}\dfrac{\mathrm{Val}_{m}^{i}\cap\mathrm{Val}_{m}^{j}}{\mathrm{Val}_{m}^{i}\cup\mathrm{Val}_{m}^{j}}, & \mathrm{Val}_{m}^{i}\cup\mathrm{Val}_{m}^{j}\neq\varnothing\\ 0, & \mathrm{Val}_{m}^{i}\cup\mathrm{Val}_{m}^{j}=\varnothing\end{cases} \tag{33}$$

To show the attribute preferences during clustering, we introduce a preference vector of attributes for each vertex in AG and denote it as $\mathbf{r}_i^{\mathrm{T}}=(r_{i1},r_{i2},...,\ r_{in_\Lambda})$ with respect to v_i. In $\mathbf{r}_i$, we have $\sum_{m=1}^{n_\Lambda}r_{im}=1$ where $r_{im}\in[0,1]$. The meaning of r_{im} is to quantify to what extent Λ_m is preferred when we determine whether another vertex v_j should be grouped in the same cluster as v_i from the attribute perspective. In other words, regarding the clustering process related to v_i, Λ_m will play a more important role if r_{im} is assigned a larger value. The preference matrix $\mathbf{R}$ is defined as $\mathbf{R}=(\mathbf{r}_1,\mathbf{r}_2,...,\mathbf{r}_{n_\mathrm{v}})^{\mathrm{T}}$.

Besides, we also have another matrix $\mathbf{D}=[d_{ij}](1\le i,j\le n_\mathrm{v})$ that is used to represent the similarity between any two vertices in AG from the topological perspective of graph. Assuming that for v_i we have $\mathrm{V}_i=\{v_k\mid e_{ik}\in\mathrm{E}\}$ representing the set of neighboring vertices of v_i, the definition of d_{ij} is given as:

$$d_{ij}=\begin{cases}\dfrac{\mathrm{V}_i\cap\mathrm{V}_j}{\mathrm{V}_i\cup\mathrm{V}_j}, & \mathrm{V}_i\cup\mathrm{V}_j\neq\varnothing\\ 0, & \mathrm{V}_i\cup\mathrm{V}_j=\varnothing\end{cases}. \tag{34}$$

Obviously the more vertices v_i and v_j have in common, the larger the value of d_{ij} is.

Given $\mathbf{T}$, $\mathbf{A}$ and $\mathbf{D}$, we aim to find appropriate $\mathbf{W}$ and $\mathbf{R}$ so that the resultant clusters can best conform to our assumptions. To solve it, we formulate an optimization problem as below,

$$\max J(\mathbf{W},\mathbf{R}) =$$
$$\partial \,\mathrm{Tr}\left((\mathbf{W_T})^{\mathrm{T}}\mathbf{W_D}\mathbf{W_T}\right) + \beta\,\mathrm{Tr}\left(\sum_{m=1}^{n_\Lambda}\mathbf{W}^{\mathrm{T}}\mathbf{S}_m\right) - \frac{\varphi}{2}\|\mathbf{W}\|_F^2 - \frac{\theta}{2}\|\mathbf{R}\|_F^2 \qquad (35)$$
$$s.t.\quad \mathbf{R1} = \mathbf{1}, \mathbf{R} \geq 0, 0 \leq \mathbf{W} \leq 1$$

where ∂, β, φ and θ are non-negative constants, $\mathbf{W_T} = \mathbf{T} \circ \mathbf{W}$ is the entrywise product of $\mathbf{T}$ and $\mathbf{W}$, $\mathbf{W_D} = \mathbf{D} \circ \mathbf{W}$ is the entrywise product of $\mathbf{D}$ and $\mathbf{W}$, $\|\mathbf{W}\|_F^2 = \mathrm{Tr}(\mathbf{W}^{\mathrm{T}}\mathbf{W})$ is the squared Frobenius norm of $\mathbf{W}$, $\|\mathbf{R}\|_F^2 = \mathrm{Tr}(\mathbf{R}^{\mathrm{T}}\mathbf{R})$ is the squared Frobenius norm of $\mathbf{R}$, $\mathbf{1}$ is a column vector with a proper size and each element of $\mathbf{1}$ is 1, and $\mathbf{S}_m = [s_{ij}^m]$ is a $n_{\mathrm{V}} \times n_{\mathrm{V}}$ matrix each cell of which is defined as:

$$s_{ij}^m = a_{ij}^m \, r_{im} \qquad (36)$$

The optimization problem as described by (35) consists of three components: a measure of clustering quality, regularizations and constraints. To clarify the eligibility of the optimization problem of (35) in terms of satisfying our purpose, we give a detailed analysis of (35) so that the eligibility can be proved.

To confirm the topological patterns of clusters identified, we constraint our analysis on the first term of (35) and rewrite it by following the trace expression as below.

$$\partial \,\mathrm{Tr}\left((\mathbf{W_T})^{\mathrm{T}}\mathbf{W_D}\mathbf{W_T}\right) = \partial \sum_{i=1}^{n_{\mathrm{V}}}\sum_{j=1}^{n_{\mathrm{V}}}\left(d_{ij}w_{ij} \times \sum_{k=1}^{n_{\mathrm{V}}} w_{ik}w_{jk}t_{ik}t_{jk}\right) \qquad (37)$$

According to the definition of $\mathbf{D}$, we know that a large value of d_{ij} indicates that v_i and v_j have a large percentage of vertices in common. For a third vertex v_k, $w_{ik}w_{jk}$ shows the degree of being grouped in the same cluster with both v_i and v_j while $t_{ik}t_{jk}$ ensures that v_k contributes to the value of $\mathrm{Tr}\left((\mathbf{W_T})^{\mathrm{T}}\mathbf{W_D}\mathbf{W_T}\right)$ only if both e_{ik} and e_{jk} are found in E. It is not difficult to conclude that if two vertices have many common vertices and most of these common vertices are also likely to be grouped in the same cluster as the two vertices we concern, the value of (37) is to be maximized. Therefore,

this conclusion, to some extent, assure that vertices in the same cluster are densely connected.

For the second term in (35), we use it to manipulate the attribute information during clustering so that clusters can be identified based on a subset of attributes with high preferences. To prove it, the second terms of (35) is rewritten with trace expression as below.

$$\beta \operatorname{Tr}(\sum_{m=1}^{n_\Lambda} \mathbf{W}^{\mathrm{T}} \mathbf{S}_m) = \beta \sum_{i=1}^{n_\mathrm{V}} \sum_{m=1}^{n_\Lambda} (r_{im} \times \sum_{j=1}^{n_\mathrm{V}} a_{ij}^m w_{ij}) \tag{38}$$

$$\beta \operatorname{Tr}(\sum_{m=1}^{n_\Lambda} \mathbf{W}^{\mathrm{T}} \mathbf{S}_m) = \beta \sum_{i=1}^{n_\mathrm{V}} \sum_{j=1}^{n_\mathrm{V}} (w_{ij} \times \sum_{m=1}^{n_\Lambda} a_{ij}^m r_{im}) \tag{39}$$

In (38) and (39), $\beta \operatorname{Tr}(\sum_{m=1}^{n_\Lambda} \mathbf{W}^{\mathrm{T}} \mathbf{S}_m)$ is expressed with two forms from the perspectives of r_{im} and w_{ij} respectively. We will take $v_i \in \mathrm{V}$ as an example to explain how r_{im} and w_{ij} are supposed to be determined in order to maximize (35). In (38), given constraints $\sum_{m=1}^{n_\Lambda} r_{im} = 1$ and $r_{im} > 0$, the preference vector of v_i, i.e., $\mathbf{r}_i$, should assign more weights (i.e., r_{im}) to attributes where large similarity scores (i.e., a_{ij}^m) between v_i and other vertices occur more frequently. The trace expression in (39) shows that w_{ij} ought to be with a large value if the amount of similarity scores between v_i and v_j (i.e., $\sum_{m=1}^{n_\Lambda} a_{ij}^m r_{im}$) is also large. In sum, combining the meanings of (38) and (39), the term $\beta \operatorname{Tr}(\sum_{m=1}^{n_\Lambda} \mathbf{W}^{\mathrm{T}} \mathbf{S}_m)$ allows us to identify clusters from a subset of attributes that are with high preference values.

After discussing the appropriateness of the first two terms of (35) as being an eligible measure of clustering quality, the other two terms in (35) are related to the regularizations of $\mathbf{W}$ and $\mathbf{R}$ respectively. For $\mathbf{W}$, we use $\frac{\varphi}{2}\|\mathbf{W}\|_F^2$ to raise the penalty

for the case that the values of all items in $\mathbf{W}$ are moving toward the maximum value (i.e., 1). The term $\frac{\theta}{2}\|\mathbf{R}\|_F^2$ is to regularize the smoothness of each preference vector in $\mathbf{R}$.

5.3.2 Details of CAP-AG

Given an AG, we will explain how the formulated problem of AG clustering, i.e., (35), can be solved in this section. First of all, the details of deriving a local optimized solution to (35) are introduced. Then we will show the complete procedure of CAP-AG.

5.3.2.1 Solution

Since the maximization problem of (35) can be considered as a quadratic optimization problem, we employ the primal-dual active set strategy introduced in [73] to search for the feasible improving directions of (35) with respect to $\mathbf{W}$ and $\mathbf{R}$ until convergence. At every iteration, CAP-AG solves a sequence of linear equality constrained quadratic subproblems with active constraints including all equality constraints and some inequality constraints that are enforced to be satisfied as equalities so that optimum $\mathbf{W}$ and $\mathbf{R}$ can be found along the boundaries of constraints. The advantage of adopting the active set strategy is that it can help solve optimization problems by only involving some subset of the variables related to active constraints and hence CAP-AG will be more efficient especially for large AGs.

For the two variable matrices $\mathbf{W}$ and $\mathbf{R}$, we adopt the strategy of optimizing $\mathbf{W}$ and $\mathbf{R}$ alternatively. That is to say, at each iteration, CAP-AG first updates $\mathbf{R}$ while keeping $\mathbf{W}$ fixed and then use the latest $\mathbf{R}$ to update $\mathbf{W}$. Assuming that we are now at the $(l+1)_{th}$ iteration with $\mathbf{W}^{(l)}$ and $\mathbf{R}^{(l)}$ available for use, the details of obtaining $\mathbf{W}^{(l+1)}$ and $\mathbf{R}^{(l+1)}$ are presented as below.

A. Updating R

To facilitate understanding, we now use $\max J(\mathbf{R}\,|\,\mathbf{W})$ to denote that $J(\mathbf{W},\mathbf{R})$ is about to be maximized by updating $\mathbf{R}$ with a fixed $\mathbf{W}$. First of all, we formulate a sequence of quadratic subproblems for approximating the maximization of $J(\mathbf{W},\mathbf{R})$ as:

$$\max J(\mathbf{R}\,|\,\mathbf{W}) = \max \sum_{i=1}^{n_{\mathrm{V}}} J(\mathbf{r}_i\,|\,\mathbf{W}) = \sum_{i=1}^{n_{\mathrm{V}}} \max J(\mathbf{r}_i\,|\,\mathbf{W}) \tag{40}$$

In (40), each subproblem is to maximize $J(\mathbf{W},\mathbf{R})$ in terms of $\mathbf{r}_i$. Therefore, the problem of updating $\mathbf{R}$ is divided into several subproblems each of which is to update the corresponding $\mathbf{r}_i$ as a part of the solution of (40).

Regarding the subproblem $\max J(\mathbf{R}\,|\,\mathbf{W})$, we rewrite it by following the standard form of quadratic problem, thus obtaining

$$\begin{aligned} &\max J(\mathbf{r}_i\,|\,\mathbf{W}) = -\frac{\theta}{2}\mathbf{r}_i^{\mathrm{T}}\mathbf{I}_{n_\Lambda}\mathbf{r}_i + \mathbf{c}_i^{\mathrm{T}}\mathbf{r}_i + b(\mathbf{W}) \\ &s.t. \quad \mathbf{1}_m^{\mathrm{T}}\mathbf{r}_i \geq 0, \quad \text{for all } 1 \leq m \leq n_\Lambda \\ &\qquad \mathbf{1}^{\mathrm{T}}\mathbf{r}_i = 1 \end{aligned} \tag{41}$$

where $\mathbf{I}_{n_\Lambda}$ is a $n_\Lambda \times n_\Lambda$ identify matrix, $\mathbf{c}_i^{\mathrm{T}} = (c_{i1}, c_{i2}, ..., c_{in_\Lambda})$, $c_{im} = \sum_{j=1}^{n_{\mathrm{V}}} w_{ij} a_{ij}^m$, $b(\mathbf{W})$ is the constant term composed of $\mathbf{W}$ but not involving $\mathbf{r}_i$ and $\mathbf{1}_m$ is a vector where the m_{th} element is 1 while the other elements are 0. For convenience, we use $\Gamma_{\geq}$ and $\Gamma_{=}$ to index $\geq$ and $=$ constraints in (41) respectively.

Regarding the setting of active constraints of (41), we define $\mathrm{P}_i^{(l+1)}$ as the set of indices of active constraints at the $(l+1)_{th}$ iteration, $n_{\mathrm{P}_i}^{(l+1)}$ as the size of $\mathrm{P}_i^{(l+1)}$, and $\mathbf{Q}_i^{(l+1)}$ as the matrix where rows are the coefficient vectors of active constraints in $\mathrm{P}_i^{(l+1)}$. Taking an arbitrary constraint $\mathbf{1}_m^{\mathrm{T}}\mathbf{r}_i \geq 0$ as an example, we add m to $\mathrm{P}_i^{(l+1)}$ if $\mathbf{1}_m^{\mathrm{T}}\mathbf{r}_i^{(l)} = 0$ and

accordingly $\mathbf{1}_m^{\mathrm{T}}$ is the corresponding row of the active constraint $\mathbf{1}_m^{\mathrm{T}}\mathbf{r}_i \geq 0$ in $\mathbf{Q}_i^{(l+1)}$. For the relationship among $\Gamma_{\geq}$, $\Gamma_{=}$ and $\mathrm{P}_i^{(l+1)}$, we have $\mathrm{P}_i^{(l+1)} \subseteq \Gamma_{\geq} \bigcup \Gamma_{=}$. Regarding the set of inactive constraints, we use $\overline{\mathrm{P}}_i^{(l+1)}$ to denote it. Since $\mathrm{P}_i^{(l+1)}$ always contains the equality $\mathbf{1}^{\mathrm{T}}\mathbf{r}_i = 1$, the size of $\overline{\mathrm{P}}_i^{(l+1)}$ is $n_{\Lambda} - n_{\mathrm{p}_i}^{(l+1)} + 1$.

Given $\mathbf{r}_i^{(l)}$, an optimal move $\Delta\mathbf{r}_i^{(l+1)}$ toward $\mathbf{r}_i^{(l+1)}$ should be able to maximize $J(\mathbf{r}_i^{(l)} + \Delta\mathbf{r}_i^{(l+1)} \mid \mathbf{W}^{(l)})$. Since active-set methods require all active constants in $\mathrm{P}_i^{(l+1)}$ to continue being active during the next move, an optimum $\Delta\mathbf{r}_i^{(l+1)}$ should solve

$$\begin{aligned} &\max J(\mathbf{r}_i^{(l)} + \Delta\mathbf{r}_i^{(l+1)} \mid \mathbf{W}^{(l)}) \\ &s.t.\ \ \mathbf{Q}_i^{(l+1)}\Delta\mathbf{r}_i^{(l+1)} = 0 \end{aligned} \tag{42}$$

The major difference between (41) and (42) is that the inequality constraints of (41) are now in equality-constraint formats in (42). To find such a $\Delta\mathbf{r}_i^{(l+1)}$, we further express the objective function (42) in its second-order Taylor representation with which we can formulate a quadratic problem for $\Delta\mathbf{r}_i^{(l+1)}$. The Taylor representation is given in (43).

$$\begin{aligned} &\max J(\mathbf{r}_i^{(l)} + \Delta\mathbf{r}_i^{(l+1)} \mid \mathbf{W}^{(l)}) \\ &= \max\left(J(\mathbf{r}_i^{(l)}|\mathbf{W}^{(l)}) + (\Delta\mathbf{r}_i^{(l+1)})^{\mathrm{T}} \cdot \nabla J(\mathbf{r}_i^{(l)}|\mathbf{W}^{(l)}) + (\Delta\mathbf{r}_i^{(l+1)})^{\mathrm{T}} \cdot \nabla^2 J(\mathbf{r}_i^{(l)}|\mathbf{W}^{(l)}) \cdot \Delta\mathbf{r}_i^{(l+1)}\right) \\ &= J(\mathbf{r}_i^{(l)}|\mathbf{W}^{(l)}) + \max\left((\Delta\mathbf{r}_i^{(l)})^{\mathrm{T}} \cdot \nabla J(\mathbf{r}_i^{(l)}|\mathbf{W}^{(l)}) + (\Delta\mathbf{r}_i^{(l+1)})^{\mathrm{T}} \cdot \nabla^2 J(\mathbf{r}_i^{(l)}|\mathbf{W}^{(l)}) \cdot \Delta\mathbf{r}_i^{(l+1)}\right) \\ &s.t.\ \ \mathbf{Q}_i^{(l+1)}\Delta\mathbf{r}_i^{(l+1)} = 0 \end{aligned} \tag{43}$$

Thus $\Delta\mathbf{r}_i^{(l+1)}$ can be solved with the corresponding Karush–Kuhn–Tucker (KKT) linear conditions of (43) as described below

$$\begin{pmatrix} \theta\,\mathbf{I}_{n_{\Lambda}} & (\mathbf{Q}_i^{(l+1)})^{\mathrm{T}} \\ \mathbf{Q}_i^{(l+1)} & 0 \end{pmatrix} \begin{pmatrix} \Delta\mathbf{r}_i^{(l+1)} \\ \boldsymbol{\varepsilon}^{(l+1)} \end{pmatrix} = \begin{pmatrix} \nabla J(\mathbf{r}_i^{(l)} \mid \mathbf{W}^{(l)}) \\ 0 \end{pmatrix} \tag{44}$$

$\boldsymbol{\varepsilon}^{(l+1)}$ in (44) is the Lagrange multiplier vector for the KTT conditions of (43). After some algebraic manipulations, $\Delta\mathbf{r}_i^{(l+1)}$ can be derived as

$$\Delta r_{im}^{(l+1)} = \begin{cases} 0, & m \in \Gamma_{\geq} \bigcap \mathrm{P}_i^{(l+1)} \\ \dfrac{1}{\theta}\left(\dfrac{\partial J(\mathbf{r}_i \mid \mathbf{W}^{(l)})}{\partial r_{im}} - \dfrac{\sum_{m \in \overline{\mathrm{P}}_i^{(l+1)}} \dfrac{\partial J(\mathbf{r}_i \mid \mathbf{W}^{(l)})}{\partial r_{im}}}{n_\Lambda - n_{\mathrm{p}_i}^{(l+1)} + 1}\right), & \text{otherwise} \end{cases} \tag{45}$$

and $\boldsymbol{\varepsilon}^{(l+1)}$ is determined by (46).

$$\varepsilon_m^{(l+1)} = \begin{cases} 0, & m \in \overline{\mathrm{P}}_i^{(l+1)} \\ \dfrac{\partial J(\mathbf{r}_i \mid \mathbf{W}^{(l)})}{\partial r_{im}} - \dfrac{\sum_{m \in \overline{\mathrm{P}}_i^{(l+1)}} \dfrac{\partial J(\mathbf{r}_i \mid \mathbf{W}^{(l)})}{\partial r_{im}}}{n_\Lambda - n_{\mathrm{p}_i}^{(l+1)} + 1}, & \text{otherwise} \end{cases} \tag{46}$$

In (45) and (46), $\dfrac{\partial J(\mathbf{r}_i \mid \mathbf{W}^{(l)})}{\partial r_{im}} = -\theta\, r_{im}^{(l)} + c_{im}^{(l)}$. Therefore, regarding the subproblem (41), we can obtain $\Delta\mathbf{r}_i^{(l+1)}$ with (45). If $\Delta\mathbf{r}_i^{(l+1)} \neq 0$, an usual update of $\mathbf{r}_i^{(l+1)}$ can be made with $\mathbf{r}_i^{(l+1)} = \mathbf{r}_i^{(l)} + \Delta\mathbf{r}_i^{(l+1)}$ so that the subproblem (41) is optimized over the active constraints. However, a full step in the direction $\Delta\mathbf{r}_i^{(l+1)}$ may cause some inactive constraints to be violated as we only consider the active constraints of (41) in (42). To avoid it, we have to find the maximum step $\lambda_i^{(l+1)}$ that we can take for the update of $\mathbf{r}_i^{(l+1)}$ in the direction $\Delta\mathbf{r}_i^{(l+1)}$. In particular, $\forall\, m \in \Gamma_{\geq}$, if $\Delta r_{im}^{(l+1)} < 0$, the condition $r_{im}^{(l)} + \lambda_{im}^{(l+1)} \cdot \Delta r_{im}^{(l+1)} \geq 0$ must be satisfied so that the update on $r_{im}^{(l+1)}$ will not violate the constraint $\mathbf{1}_m^{\mathrm{T}}\mathbf{r}_i \geq 0$. For $\lambda_i^{(l+1)}$, we can determine it as:

$$\lambda_i^{(l+1)} = \min\left\{1,\ \min\left\{\lambda_{im}^{(l+1)} = \frac{r_{im}^{(l)}}{-\Delta r_{im}^{(l+1)}} : m \in \Gamma_{\geq}, \Delta r_{im}^{(l+1)} < 0\right\}\right\} \tag{47}$$

where 1 accounts for the equality constraint in (41). $\lambda_i^{(l+1)}$, together with $\mathbf{r}_i^{(l)}$ and $_{\Delta}\mathbf{r}_i^{(l+1)}$, is then used to obtain $\mathbf{r}_i^{(l+1)}$ according to (48).

$$\mathbf{r}_i^{(l+1)} = \mathbf{r}_i^{(l)} + \lambda_i^{(l+1)} \cdot {}_{\Delta}\mathbf{r}_i^{(l+1)} \tag{48}$$

Once we have obtained $\mathbf{r}_i^{(l+1)}$ for each $v_i \in \mathrm{V}$, $\mathbf{R}^{(l+1)}$ can be determined. The next step is to get $\mathbf{W}^{(l+1)}$ given $\mathbf{R}^{(l+1)}$.

B. Updating W

Similar to the update of $\mathbf{R}$, we use $\max J(\mathbf{R}\,|\,\mathbf{W})$ to represent the optimization problem of $\max J(\mathbf{R}\,|\,\mathbf{W})$ in terms of $\mathbf{W}$ by fixing $\mathbf{R}$. Observing (35), we find that each element of $\mathbf{W}$ is independent with others as there are no constraints between any two elements in $\mathbf{W}$. Therefore, we can approximate $\max J(\mathbf{R}\,|\,\mathbf{W})$ as follows

$$\begin{aligned} &\max J(\mathbf{W}\,|\,\mathbf{R}) = \max \sum_{i=1}^{n_{\mathrm{V}}} \sum_{j=1}^{n_{\mathrm{V}}} J(w_{ij}\,|\,\mathbf{R}) = \sum_{i=1}^{n_{\mathrm{V}}} \sum_{j=1}^{n_{\mathrm{V}}} \max J(w_{ij}\,|\,\mathbf{R}) \\ &s.t.\ \ 0 \le w_{ij} \le 1 \end{aligned} \tag{49}$$

where $J(w_{ij}\,|\,\mathbf{R})$ is given in (50).

$$\begin{aligned} J(w_{ij}\,|\,\mathbf{R}) &= -\frac{\varphi}{2} w_{ij}^2 + \partial\, d_{ij} w_{ij} \sum_{k=1}^{n_{\mathrm{V}}} w_{ik} w_{jk} t_{ik} t_{jk} + \beta\, w_{ij} \sum_{m=1}^{n_{\Lambda}} s_{ij}^m \\ &= -\frac{\varphi}{2} w_{ij}^2 + w_{ij} (\partial\, d_{ij} \sum_{k=1}^{n_{\mathrm{V}}} w_{ik} w_{jk} t_{ik} t_{jk} + \beta \sum_{m=1}^{n_{\Lambda}} s_{ij}^m) \end{aligned} \tag{50}$$

Note that the coefficient $d_{ij} w_{jk} t_{ij} t_{jk}$ will become a part of the coefficient of w_{ij}^2 if $k = j$; however, since the self-connection of vertex is not considered for AG clustering, we have $t_{jj} = 0$ and accordingly $d_{ij} w_{jk} t_{ij} t_{jk} = 0$ when $k = j$. In this regard, it is not necessary to consider the case of $k = j$ separately in (50), so we still reckon $d_{ij} w_{jk} t_{ij} t_{jk}$ as a part of the coefficient of w_{ij} when $k = j$ for simplicity.

From (50), the problem of $\max J(\mathbf{R}\,|\,\mathbf{W})$ is converted into a sequence of subproblems in terms of w_{ij}. In fact, the subproblem $\max J(w_{ij}\,|\,\mathbf{R})$ is essentially a maximization issue as indicated by (50). Because $\frac{d^2 J(w_{ij}\,|\,\mathbf{R})}{d\,w_{ij}^{\,2}} = -\varphi < 0$, (50) is a concave function with respect to w_{ij}. It is easy to conclude that the maximum value of (50) will be obtained when $\frac{d\,J(w_{ij}\,|\,\mathbf{R})}{d\,w_{ij}} = 0$ if without the constraint $0 \le w_{ij} \le 1$. Assuming that w_{ij}^* is the value of w_{ij} that satisfies the equation $\frac{d\,J(w_{ij}\,|\,\mathbf{R}^{(l+1)})}{d\,w_{ij}} = 0$, we have

$$w_{ij}^* = \frac{1}{\varphi}\left(\partial\, d_{ij} \sum_{k=1}^{n_{\mathrm{V}}} w_{ik}^{(l)} w_{jk}^{(l)} t_{ik} t_{jk} + \beta \sum_{m=1}^{n_{\Lambda}} a_{ij}^{m} r_{im}^{(l+1)} \right). \tag{51}$$

Therefore at $(l+1)_{th}$ iteration, the solution to the subproblem $\max J(w_{ij}^{(l+1)}\,|\,\mathbf{R}^{(l+1)})$ when considering the constraint $0 \le w_{ij}^{(l+1)} \le 1$ is given in (52).

$$w_{ij}^{(l+1)} = \begin{cases} 0, & w_{ij}^* \le 0 \\ w_{ij}^*, & 0 < w_{ij}^* < 1 \\ 1, & w_{ij}^* \ge 1 \end{cases} \tag{52}$$

So far, we have successfully derived $\mathbf{R}^{(l+1)}$ and $\mathbf{W}^{(l+1)}$ with (51) and (52) respectively at $(l+1)_{th}$ iteration. Based the relationships among $\mathbf{R}^{(l+1)}$, $\mathbf{W}^{(l+1)}$, $\mathbf{R}^{(l)}$ and $\mathbf{W}^{(l)}$, we propose CAP-AG to find the optimal solution of $\max J(\mathbf{R}\,|\,\mathbf{W})$, thus identifying C.

5.3.2.2 CAP-AG

CAP-AG adopts an iteration procedure to search for a local optimal solution of $\max J(\mathbf{W},\mathbf{R})$. At the $(l+1)_{th}$ iteration, the previous results of $\mathbf{R}$ and $\mathbf{W}$, i.e., $\mathbf{R}^{(l)}$ and $\mathbf{W}^{(l)}$, will be used to reestimate $\mathbf{R}^{(l+1)}$ and $\mathbf{W}^{(l+1)}$ based on the formulas devised in the last section. The iteration procedure will be terminated if a convergence of

$\max J(\mathbf{W},\mathbf{R})$ is reached or the procedure is now at the maximum number of iterations $l_{\max}$. Regarding the convergence of $\max J(\mathbf{W},\mathbf{R})$, the difference between $J(\mathbf{W}^{(l+1)},\mathbf{R}^{(l+1)})$ and $J(\mathbf{W}^{(l)},\mathbf{R}^{(l)})$ should not be more than a predefined threshold, i.e., δ.

We present the complete procedure of CAP-AG in Figure 11. Lines 4-19 are to describe how we iteratively find the optimum $\mathbf{R}$ and $\mathbf{W}$ to maximize $J(\mathbf{W},\mathbf{R})$. It is noted that we check the signs of each element in $\boldsymbol{\varepsilon}^{(l+1)}$ as indicated by line 8. The reason we perform this check is ascribed to the sign restrictions of KTT conditions we use to obtain $_{\Delta}\mathbf{r}_i^{(l+1)}$. In particular, to satisfy the sign restrictions of (44), each Lagrange multiplier $\varepsilon_m^{(l+1)}$ in $\boldsymbol{\varepsilon}^{(l+1)}$ should meet the requirement $\varepsilon_m^{(l+1)} \le 0$. However, if $\forall \varepsilon_m^{(l+1)} \in \boldsymbol{\varepsilon}^{(l+1)} : \varepsilon_m^{(l+1)} > 0$, it means that the corresponding active constraint indexed with m must be allowed to become a strict inequality. Thus we remove this constraint from $\mathrm{P}_i^{(l+1)}$ so that the requirement of keeping active will not be applied to it and then repeat the update procedure of $\mathbf{Q}_i^{(l+1)}$. Lines 20-24 is to prune the original AG by removing edges whose incident vertices are not likely to be grouped in the same cluster according to $\mathbf{W}$. Line 25 shows the minimum size of cluster we want to obtain. In line 26, we consider each connected part in AG after pruning as a cluster. Therefore, after adding all connected parts of AG to C, we finally obtain the resultant clusters identified by CAP-AG from the given AG.

Algorithm: CAP-AG

Input: AG , ∂ , β, φ, θ, l_{max} , δ , w_{min}

Output: C

1: obtain $\mathbf{T}$, $\mathbf{A}$ and $\mathbf{D}$ with (32), (33) and (34) respectively
2: randomly initialize $\mathbf{R}^{(0)}$ and $\mathbf{W}^{(0)}$ by following the constraints in (35)
3: $l = 0$
4: **repeat**
5: **for** each $v_i \in \mathrm{V}$
6: update $\mathrm{P}_i^{(l+1)}$ and $\mathbf{Q}_i^{(l+1)}$ based on $\mathbf{R}^{(l)}$
7: obtain $\Delta\mathbf{r}_i^{(l+1)}$ and $\boldsymbol{\varepsilon}^{(l+1)}$ with (45) and (46) respectively
8: **if** $\Delta\mathbf{r}_i^{(l+1)} = 0$ and $\forall \varepsilon_m^{(l+1)} \in \boldsymbol{\varepsilon}^{(l+1)} : \varepsilon_m^{(l+1)} > 0$
9: remove the constraint indexed with m from $\mathrm{P}_i^{(l+1)}$
10: update $\mathbf{Q}_i^{(l+1)}$ accordingly
11: go to line 7
12: **end if**
13: obtain $\lambda_i^{(l+1)}$ computed as (47)
14: obtain $\mathbf{r}_i^{(l+1)}$ with (48)
15: **end for**
16: update $\mathbf{W}^{(l+1)}$ with (52)
17: $\delta^{(l+1)} = \left| J(\mathbf{W}^{(l+1)}, \mathbf{R}^{(l+1)}) - J(\mathbf{W}^{(l)}, \mathbf{R}^{(l)}) \right|$
18: $l = l + 1$
19: **until** $\delta^{(l)} \le \delta$ or $l > l_{\max}$
20: **for** each $e_{ij} \in \mathrm{E}$
21: **if** $w_{ij} < w_{min}$ or $w_{ji} < w_{min}$
22: remove e_{ij} from E
23: **end if**
24: **end for**
25: remove all vertices that have no edges in E
26: connected parts in AG are considered as clusters in C
27: **return** C

Figure 11. The complete procedure of CAP-AG.

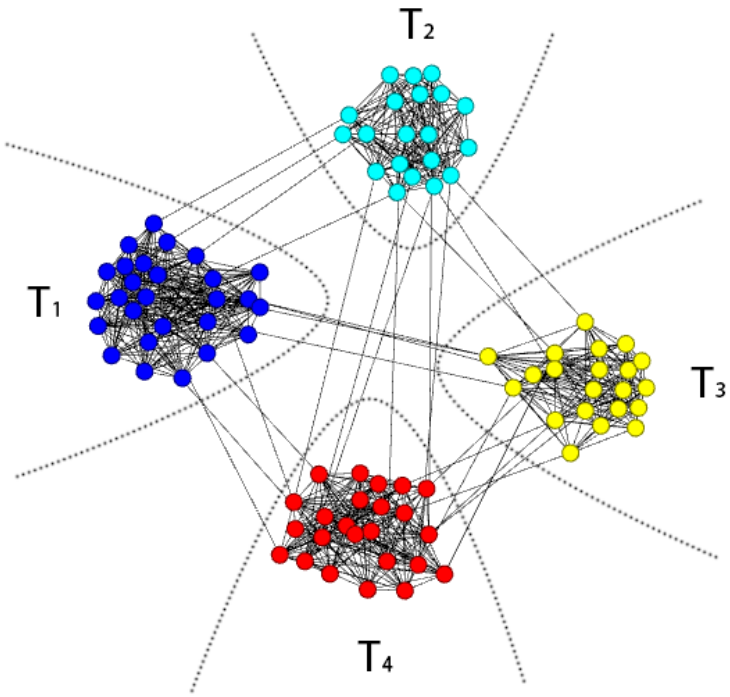

Figure 12. The AG used in the running example. Four clusters are generated and they are highlighted by different colors.

5.3.3 Running Example

In the running example, we apply CAP-AG to a toy problem of AG clustering. The simulated graph data, as shown in Figure 12, contains 91 vertices and 753 edges. We artificially create four clusters $\mathrm{T}=\{\mathrm{T}_f\}(1\leq f\leq 4)$ and use different colors to highlight them in Figure 12. Regarding graph topology, it is observed that vertices in the same cluster are densely connected while few vertices from different clusters are connected. For the attribute information of vertices, there are five attributes that can be associated with vertices and thus we have $\Lambda=\{\Lambda_m\}(1\leq m\leq 5)$. In this running example, different attributes are preferred by vertices in different clusters. To clarify it, we present the distributions of Λ_1, Λ_2, Λ_3, Λ_4 and Λ_5 in Figure 13(a)-(e) respectively. Taking the distribution of Λ_1 in Figure 13(a) as an example, we can see that vertices in the cluster T_1 are much more similar in Λ_1 as indicated by the large

values of a_{ij}^{1}. In other words, for vertices in cluster T_1, the values of Λ_1 are more frequently found than those of other attributes. In addition, to testify the robustness of CAP-AG, we introduce the attribute Λ_5 whose values are randomly assigned to all vertices with the same probability. In Figure 13(e), there is no notable preference about taking values of Λ_5 for vertices in different clusters, so we consider it as the noisy attribute.

To measure the degree of matching between the clusters identified and the ground truth, the metric *NMI* used in 4.5.1 is adopted.

Before applying CAP-AG to the running example, we need to determine ∂, β, φ and θ in advance. After analyzing the graph data, we have $d_{avg} = 0.13$, $a_{avg} = 0.18$ and $t_{avg} = 2.95$. Therefore, according to the suggestions given in 5.3.2, we set both of ∂ and β equal to 1 so that $\partial\, d_{avg} t_{avg}$ and $\beta\, a_{avg}$ have the same magnitude. For φ and θ, we set their values to 1 as well. Regarding the values of other parameters, we set them as $l_{max} = 100$, $\delta = 1$ and $w_{min} = 0.5$.

Starting with a random initialization of $\mathbf{W}$ and $\mathbf{R}$, CAP-AG eventually obtains a set of clusters, i.e., C, as well as the optimized $\mathbf{W}$ and $\mathbf{R}$. To compare C with the ground truth, we compute the score of *NMI* and find that the result of *NMI* is 1. That is to say, the clusters in C are identical to the ground truth of clusters. Furthermore, we present the preference vector of each vertex in Figure 13(f) according to the optimum $\mathbf{R}$. Note that vertices along v_i-axis are highlighted with different colors that are consistent with the colors of clusters in Figure 12 and therefore the clusters they belong to can be indicated. From Figure 13(f), it is seen that the preference vectors of vertices clearly state the dependencies between attributes and clusters. For example, in Figure 13(f), blue vertices along v_i-axis belong to the cluster T_1 according to Figure 12 and all of them have the same preference vector $(1, 0, 0, 0, 0)$, which in return

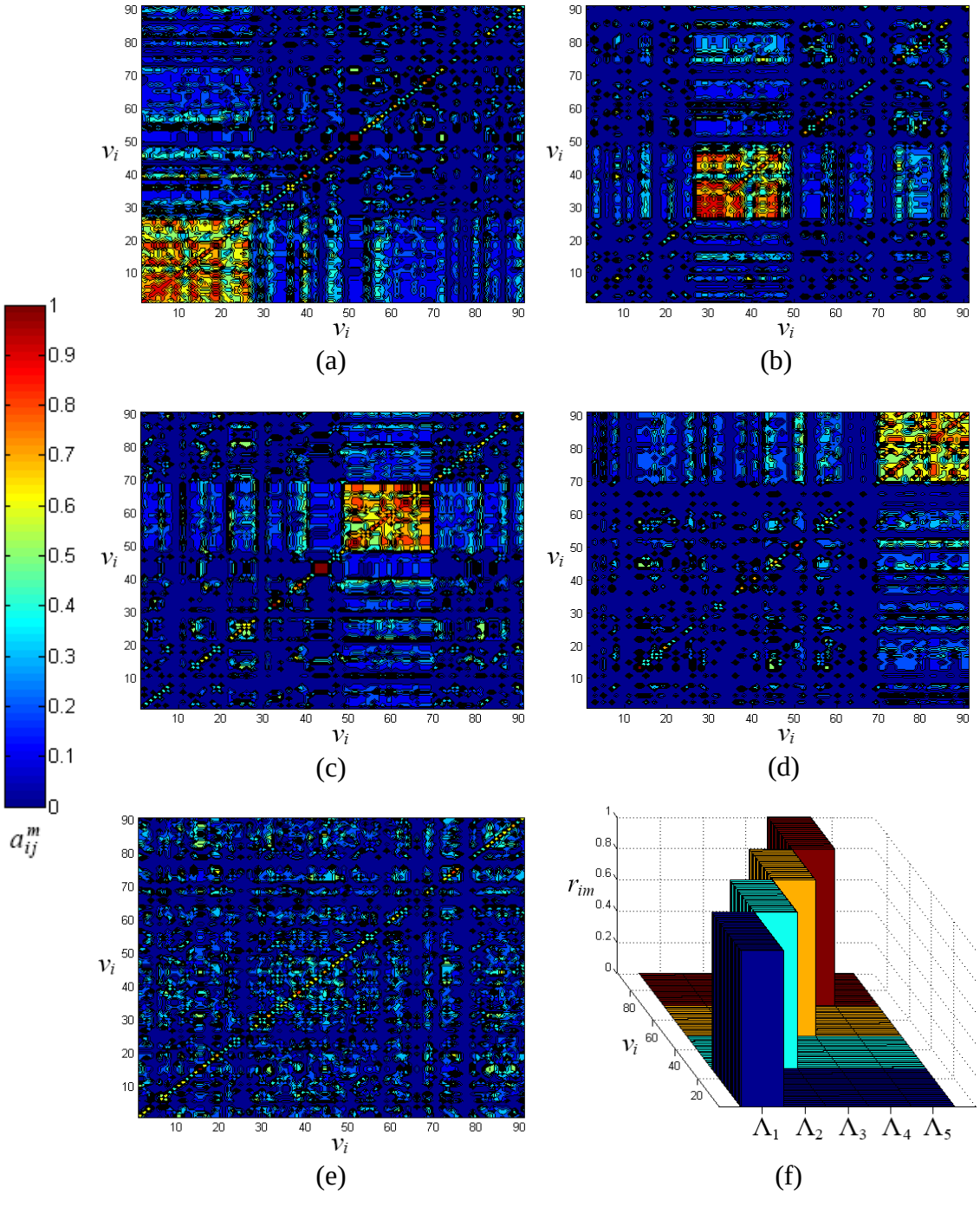

Figure 13. (a)-(e) are the distributions of a_{ij}^m for the attributes Λ_1, Λ_2, Λ_3, Λ_4 and Λ_5, (f) is the distribution of the optimized **R**.

discloses the strong dependency between T_1 and Λ_1. Therefore, we have reason to believe that Λ_1 is of significance to the study of T_1. Lastly, the influence of Λ_5 as a

noisy attribute is small enough to be ignored by CAP-AG as the preference values of Λ_5 in all preference vectors are 0. In sum, a conclusion can be made that the clustering results obtained by CAP-AG are consistent with our assumptions and the ability of disclosing the dependencies between clusters and attributes has been confirmed.

5.4 Application of Identifying Protein Complexes

5.4.1 Background

A protein complex is a biomolecule that contains a number of proteins interacting with each other to perform different cellular functions [80]. For example, the *exosome complex* is capable of degrading various types of RNA molecules and the *nuclear pore complex*, probably the largest complex in a cell, is responsible for the protected exchange of components between the nucleus and the cytoplasm as well as for preventing the transport of materials that are not allowed to cross the nuclear envelope.

The identification of protein complexes in a PPIN can, therefore, lead to a better understanding of the roles of such a network in different cellular systems and it is, for this reason, that the protein complex identification problem has received a lot of attentions and a considerable number of techniques, including both laboratory based and computational techniques, have been proposed to address the problem.

To identify protein complexes on a large scale, many laboratory experiments involving affinity purification followed by mass spectrometry (AP/MS) have to be carried out [32], [39]. For example, given a fusion gene containing a bait protein and a chemical tag with high affinity, AP/MS has to be used to purify all proteins and identify from among them those that are bound to the bait protein as these are the proteins that are considered as making up a protein complex. After these proteins are identified, mass spectrometry methods are then used again to characterize each of them. Though effective, AP/MS is not a very efficient approach as many experiments, using different

bait proteins every time, have to be carried out [23]. Also, the set of protein complexes that can be identified within a PPI network using such laboratory approach is usually incomplete, as some protein complexes may not be discoverable under certain experimental conditions [51].

To avoid the problems associated with laboratory identification of protein complexes, a number of different computational approaches have recently been proposed to automate the task. Such approaches may not be feasible in the past but with the advent of high-throughput methods, such as the yeast two-hybrid systems [41], it is now possible for large numbers of PPI networks to be constructed and made available for these computational approaches to be tested with.

Even though there are quite a number of different computational approaches, they are, by and largely, developed based on the use of different graph clustering algorithms. By representing a PPI network as a graph whose vertices represent proteins and edges are interactions between proteins, these algorithms can discover graph clusters based on different topological properties, such as density, k-cores, core-attachment structures and peripheries. However, among these topological properties used for graph clustering, density is the most commonly used as there is some evidence from known PPI networks that proteins in a protein complex tend to interact more with each other in the same complex, thus forming a more densely interacting region within a PPI network [87]. All other topological properties used for graph clustering are therefore direct or indirect derivations from the density measures. Additional requirement on the structures of the graph clusters discovered is taken into consideration whenever necessary.

One of the most popular graph clustering algorithms that can identify dense clusters effectively and that has been used quite frequently for protein complex identification is the MCL algorithm [27]. The MCL algorithm has been shown to be relatively accurate and robust [51], [13]. As mentioned in 2.2.2, MCL discovers dense clusters in a graph by simulating flow expansion and contraction. With MCL, a set of dense

clusters can then be extracted as protein complexes from the incidence matrix of a PPI network graph. A major weakness of MCL is that it is not capable of taking into consideration biological information while a graph is expanded or inflated. In other words, it can only consider topological information when searching for protein complexes in a PPI network.

Apart from MCL, other graph clustering algorithms have also been used to identify protein complexes. In [6], for example, MCODE is used to detect densely connected regions in a PPI network by taking into consideration local neighborhood density using vertex weighting as opposed to transition matrices. As another example, RNSC [45] is used to identify protein complexes through graph partitioning. By defining several cost functions based on cluster size, cluster density and functional homogeneity, RNSC finds an optimal partitioning of the vertices in a PPI network graph so that each partition may represent a protein complex.

Similar to RNSC, the approach proposed by Ding et al. [26] also attempts to partition a PPI network graph except that it makes use of a minimum vertex-cut to identify boundaries of graph clusters as protein complexes that are relatively denser than their nearby regions.

In [4], a graph mining algorithm called DPClus is proposed to refine graph clusters discovered by keeping track of cluster periphery through the monitoring of cluster properties. The efficiency of DPClus is improved by another algorithm called IPCA [49]. IPCA considers a combination of vertex distance and subgraph density when performs graph clustering.

Another algorithm for graph clustering that is based on a different idea than IPCA is called CFinder [1]. It is used to identify protein complexes that are characterized by k-clique structure in a PPI network. In [52], a graph clustering approach that makes use of cliques for the finding of protein complexes is also proposed. As it is believed that there is a relatively high false positive probability for the interactions found in PPI

networks through high-throughput experiments, an iterative scoring method is introduced in [52] to assign weights that can reflect their reliability to protein interactions. Based on these weights, an algorithm called CMC is used to identify protein complexes based on the identification of maximal cliques.

In [93], another algorithm called COACH that considers a different set of topology properties is proposed by Wu et al. COACH takes advantage of the core-attachment structure of protein complexes to detect for protein complexes in a PPI network.

Not many graph clustering algorithms can discover overlapping protein complexes. The algorithm proposed in [102] is an exception. It uses a regularized sparse generative network model to detect for overlapping protein complexes. Based on such a model, peripheral proteins that have few links with the cores of protein complexes are specially handled based on the use of propensities generated by exponential distribution.

Recently, some algorithms (e.g. [48] and [50]) have been developed to identify protein complexes based on attribute information about proteins rather than on topological properties of PPI networks. This is because, other than some evidence pointing to protein complexes having special network topologies, there has also been some recent evidence that proteins within the same PPI complex may perform the same functions and it is probably for this reason that at least one algorithm is developed to identify protein complexes based on the similarity of protein functions [48]. There has also been some attempt to use gene expression data together with PPI networks to discover protein complexes [50].

To evaluate the performance of MCL-AG and CAP-AG, we apply them to identify protein complexes and compare with the popular approaches including PCIA，MCL, MCODE, RNSC, DPClus, IPCA, CFinder, CMC and COACH. The experiment results will be presented and analyzed in the next section.

5.4.2 Experimental Results and Discussions

5.4.2.1 Data Preparation

For the purpose of performance evaluation, MCL-AG and CAP-AG have been tested with three sets of real PPI network data, including *Krogan 2006* [46], *DIP Scere* [94] and *DIP Hsapi* [94]. *Krogan 2006* is related to yeast Saccharomyces Cerevisiae. It can be obtained from the BioGRID database [81]. In BioGRID, there are two sets of *Krogan 2006* data. For our experiments, we used the *core* set of *Krogan 2006* where the PPI data is obtained with a higher reliability than its *extended* data set [46]. *DIP Scere* is also collected from Saccharomyces Cerevisiae but it is much larger in size than that of *Krogan 2006* (see Table 11). Unlike these two data sets, *DIP Hsapi* is collected from human beings. The PPI networks of *DIP Scere* and *DIP Hsapi* used for our experiments were obtained from the snapshot of the DIP database [94] as of August 18, 2012, which was the latest version that we could obtain when we worked on our experiments. Before we used the data sets, we removed all self-connecting and duplicated interactions from the data sets. After the removal of such interactions, the characteristics of all three PPI network data sets used in our experiments are given in Table 11.

The Gene Ontology (GO) project [5] was utilized as the attribute information of proteins. According to the GO project, there are three attributes of proteins corresponding to the three GO categories of *biological processes, molecular functions* and *cellular components* and they are represented as Λ_p, Λ_f and Λ_c respectively. In particular, *biological processes* are concerned with certain biological objectives with which a protein is involved in; *molecular functions* refer to the biochemical activities that a protein performs; *cellular components* are concerned with the locations where a protein is active in the cells. For our experiments, the attributes of the proteins in the three data sets were obtained from the GO databases [14]. GO terms in the category

of *cellular component* that may give any hints as to what complex(es) a protein may belong to had been excluded from our experiments.

Table 11. Statistics of PPI Networks

PPI Network	Number of Proteins	Number of Interactions		
		Before	After	% Difference
Krogan 2006	2674	7079	7075	-0.06%
DIP Scere	4584	21161	20845	-1.49%
DIP Hsapi	2523	3369	3053	-9.38%

To evaluate the performances of all algorithms, we compared the protein complexes they identified with the known protein complexes contained in the MIPS/CYGD [35] and CYC2008 [72] databases as of March 11, 2013. The complexes in MIPS/CYGD belong to Sacchromyces Cerevisiae and there are altogether 255 known protein complexes. In addition to the protein complexes in MIPS/CYGD, we also compared what PCIA and other algorithms discovered with the known protein complexes in CYC2008 for Sacchromyces Cerevisiae. There CYC2008 contains 408 such known complexes. By merging MIPS/CYGD and CYC2008, we obtained a total of 557 known protein complexes for Sacchromyces Cerevisiae for comparison.

For *DIP Hsapi*, what were identified by the proposed approaches and other algorithms were compared against those known complexes in the MIPS/CORUM [76] database. In this database, there are altogether 2835 known protein complexes. It should be noted that, since the databases of both PPI networks and known protein complexes are constantly being updated, the experiment results obtained with older or newer versions of the data may not be the same when the experiments described in this section are repeated with them.

5.4.2.2 Experiment Setup

For performance evaluation, MCL-AG, CAP-AG and other algorithms including MCL, MCODE, RNSC, DPClus, IPCA, CFinder, CMC and COACH were used to identify protein complexes in the three data sets as described above. The parameter settings for the algorithms that we used for testing are listed in Table 12.

Table 12. Parameter Settings of Algorithms used in Our Experiments

Algorithm	**Parameter Setting**
MCL-AG	inflation = 1.8, $\mu = 0.8$
CAP-AG	$\partial = 0.5$, $\beta = 0.5$, $\varphi = 1$, $\theta = 1$
MCL	inflation = 1.8
MCODE	VWP = 0.2, haircut = TRUE, max depth = 100, fluff = FALSE, node score cutoff = 0.2
RNSC	N/A
DPClus	$d_{in} = 0.6$, $cp_{in} = 0.5$
IPCA	S = 2, P = 2, $T_{in} = 0.6$
CFinder	N/A
CMC	min_size = 2, overlap_thres = 0.5, merge_thres = 0.25
COACH	redundancy threshold = 0.225

All these parameters were set based either on the recommendations of the respective authors or through a series of trials and errors we performed. For the latter, the best parameter settings were found by finding the parameter sets that gave the best average results of the evaluation measures that we used in performance analysis for all three PPI network data sets that we described above.

Taking the algorithm IPCA as an example, to find the best setting of T_{in} we tried setting it from 0.1 to 1.0, both inclusive, increasing at the step of 0.1 each time. After the trials and errors, we found that when T_{in} was set to 0.6, IPCA performed the best in terms of the average performance of the two indicators, i.e., the matching rate and

f-measure, that we used in our experiments. We noticed that although many previous works suggested different best values of T_{in} of IPCA, such as 0.4 in [87] and 0.9 in [13], in our case, $T_{in} = 0.6$ gave us the best performance for IPCA. In fact, when it was set to 0.6, IPCA performed much better than when T_{in} was set to 0.9 or 0.4 in our experiments. Hence, for our experiments T_{in} was set to 0.6 (Table 12).

To ensure that there were no biases, we used the same inflation parameter value for both MCL-AG and MCL, in our experiments. Regarding CAP-AG, we chose the values of ∂, β, φ and θ among $\partial, \beta \in \{0.25, 0.5, 0.75\}$ and $\varphi, \theta \in \{0.5, 1\}$ based on the trials we performed. We noted that the performance of our approach did not change much with the values of ∂, β, φ and θ. Setting $\partial = 0.5$, $\beta = 0.5$, $\varphi = 1$ and $\theta = 1$ gave reliable performances in most trials.

5.4.2.3 Evaluation Metrics

Regarding the evaluation metrics, we adopted two metrics, the matching-rate (*mr*) measure and *f*-measure, to perform the comparison. In particular, the *mr* measure is to compute the degrees of matches between the complexes identified by each of these algorithms and those that are previously known and it is defined as:

$$mr = \max_{f_2}\left(\frac{n_{\mathrm{C}_{f1},\mathrm{T}_{f2}}}{n_{\mathrm{C}_{f1}} n_{\mathrm{T}_{f2}}}\right) \tag{53}$$

where $n_{\mathrm{C}_{f1}}$ is the number of proteins in $\mathrm{C}_{f1} \in \mathrm{C}$, $n_{\mathrm{T}_{f2}}$ is the number of proteins in $\mathrm{T}_{f2} \in \mathrm{T}$, and $n_{\mathrm{C}_{f_1},\mathrm{T}_{f_2}}$ is the number of proteins that are found in both C_{f_1} and T_{f_2}. This mr measure can be considered as a measure of the maximum overlap between an identified complex (i.e., C_{f_1}) and a known complex that it matches with (i.e., T_{f_2}). Since the matching rate takes on a value within the range of 0 and 1, we can define a ρ as the threshold of a minimum-acceptable mr. For the purpose of performance

analysis, this minimum *mr* was set to $\rho \geq 0.7$, $\rho \geq 0.8$, $\rho \geq 0.9$ and $\rho = 1$ for each algorithm so that their performance can be more easily compared.

To determine quality of the protein complexes identified, in addition to *mr*, we also used the *f*-measure as a measure of the overall accuracy of each algorithm. Assuming that an identified protein complex is matched against a known complex in the MIPS database and ρ is set as $\rho \geq 0.2$, then the *f*-measure can be defined as:

$$f\text{-measure} = 2 \times \frac{precision \times recall}{precision + recall}. \quad (54)$$

In (54), $precision = n_{C,\rho \geq 0.2} / n_{\mathrm{C}}$, where $n_{C,\rho \geq 0.2}$ is the number of identified protein complexes whose matching rates are not less than 0.2, and $recall = n_{T,\rho \geq 0.2} / n_{\mathrm{T}}$, where n_{T} is the size of the set of known complexes and $n_{T,\rho \geq 0.2}$ is the number of known protein complexes that are correctly identified with matching rates not less than 0.2.

5.4.2.4 Experimental Results

A. Overall Evaluation

For performance comparison, the protein complexes identified by each of MCL-AG, CAP-AG, MCL, MCODE, RNSC, DPClus, IPCA, CFinder, CMC and COACH in the PPINs *Krogan 2006*, *DIP Scere* and *DIP Hsapi* were compared with those known complexes in T. The results in terms of the *f*-measure and the *mr* are presented in Tables 13 to 18 respectively. The symbol, #, in the Tables 16-18 denotes the total number of protein complexes identified.

Table 13. The *f*-measure scores for *Krogan 2006*

	Precision	***Recall***	***f*-measure**
MCL-AG	0.34	$0.66^{(1st)}$	$0.45^{(2nd)}$
CAP-AG	0.57	0.44	$0.5^{(1st)}$

MCL	0.33	0.46	0.38
MCODE	0.7(2nd)	0.16	0.26
RNSC	0.33	0.57(2nd)	0.42
DPClus	0.38	0.55(3rd)	0.45(2nd)
IPCA	0.26	0.52	0.35
CFinder	0.73(1st)	0.31	0.44(3rd)
CMC	0.44	0.34	0.38
COACH	0.59(3rd)	0.33	0.42

Table 14. The *f*-measure scores for *DIP Scere*

	Precision	***Recall***	***f*-measure**
MCL-AG	0.26	0.72(1st)	0.38(3rd)
CAP-AG	0.42(3rd)	0.35	0.38(3rd)
MCL	0.44(2nd)	0.09	0.15
MCODE	0.22	0.67(2nd)	0.33
RNSC	0.25	0.52	0.34
DPClus	0.19	0.65(3rd)	0.29
IPCA	0.58(1st)	0.43	0.49(1st)
CFinder	0.29	0.56	0.38(3rd)
CMC	0.39	0.52	0.45(2nd)
COACH	0.23	0.45	0.3

Table 15. The *f*-measure scores for *DIP Hsapi*

	Precision	***Recall***	***f*-measure**
MCL-AG	0.36	0.3(1st)	0.33(1st)
CAP-AG	0.41	0.11	0.17
MCL	0.3	0.2	0.24(3rd)
MCODE	0.49	0.05	0.09
RNSC	0.33	0.25(2nd)	0.28(2nd)
DPClus	0.29	0.21	0.24(3rd)

IPCA	0.19	0.23(3rd)	0.21
CFinder	0.64(2nd)	0.13	0.22
CMC	0.58(3rd)	0.13	0.21
COACH	0.67(1st)	0.14	0.23

Table 16. The *mr* scores for *Krogan 2006*

	#	$\rho \geq 0.7$	$\rho \geq 0.8$	$\rho \geq 0.9$	$\rho = 1$
MCL-AG	1823	74(2nd)	59(2nd)	49(2nd)	47(2nd)
CAP-AG	276	77(1st)	64(1st)	52(1st)	51(1st)
MCL	545	47	37	27(3rd)	26
MCODE	71	22	17	11	11
RNSC	752	72	55(3rd)	45(3rd)	43(3rd)
DPClus	599	67	54	41	40
IPCA	1873	50	21	0	0
CFinder	261	78(3rd)	49	23	21
CMC	297	47	35	23	21
COACH	347	60	38	19	17

Table 17. The *mr* scores for *DIP Scere*

	#	$\rho \geq 0.7$	$\rho \geq 0.8$	$\rho \geq 0.9$	$\rho = 1$
MCL-AG	1823	74(1st)	59(1st)	49(1st)	47(1st)
CAP-AG	347	54(2nd)	49(2nd)	46(2nd)	44(2nd)
MCL	834	24	19	16	15
MCODE	62	11	6	2	1
RNSC	1392	46	35(3rd)	27(3rd)	26(3rd)
DPClus	880	26	15	10	10
IPCA	3682	29	13	1	0
CFinder	427	54(2nd)	32	18	17
CMC	1152	47(3rd)	29	23	23
COACH	853	46	27	15	14

Table 18. The *mr* scores for *DIP Hsapi*

	#	$\rho \geq 0.7$	$\rho \geq 0.8$	$\rho \geq 0.9$	$\rho = 1$
MCL-AG	855	$26^{(1st)}$	$20^{(1st)}$	$18^{(2nd)}$	$18^{(2nd)}$
CAP-AG	179	18	$16^{(3rd)}$	$13^{(3rd)}$	$13^{(3rd)}$
MCL	556	18	15	$13^{(3rd)}$	$13^{(3rd)}$
MCODE	69	3	2	0	0
RNSC	738	$22^{(3rd)}$	$20^{(1st)}$	$20^{(1st)}$	$20^{(1st)}$
DPClus	565	$23^{(2nd)}$	$17^{(2nd)}$	$13^{(3rd)}$	$13^{(3rd)}$
IPCA	1733	2	1	0	0
CFinder	134	10	6	4	3
CMC	136	13	7	4	4
COACH	150	13	6	4	4

In Tables 13-15, we present detailed results showing the *precision*, *recall* and the *f*-measure scores calculated for all algorithms tested. Among these algorithms, both MCL-AG and CAP-AG have a promising performance in terms of *f*-measure. In particular, MCL-AG outperforms the other algorithms when applied to all PPINs and CAP-AG obtains a better performance than others when applied to the yeast PPINs, *Krogan 2006* and *DIP Scere*. Hence, we believe that both MCL-AG and CAP-AG are more capable of identifying protein complexes.

When comparing MCL-AG and CAP-AG from Tables 13-15, we find that MCL-AG performs better than CAP-AG in terms of *recall* while CAP-AG performs better than MCL-AG in terms of *precision*. This is because the total number of different proteins found in the manually curated protein complexes represents only a relatively small portion of the total number of different proteins found in the whole PPI network. For example, the total number of different proteins found in the protein complexes in MIPS/CYGD and CYC2008 represent only a small portion of the total number of different proteins found in the PPI networks of *Krogan 2006* and *DIP Scere*. In fact, only half of the total number of different proteins in *Krogan 2006* and one third of that

in *DIP Scere* are found in the known protein complexes. Hence, algorithms that detect less protein complexes are more likely to achieve higher scores in the measure of *precision* while the algorithms that detect more protein complexes are more likely to achieve higher scores in the measure of *recall.*

When it comes to the performance of the different algorithms in terms of the *matching rates* between computationally identified complexes and those that are previously known, the results are shown in Tables 16-18.

For all of *Krogan 2006*, *DIP Scere*, and *DIP Hsapi*, MCL-AG is found to have performed better than other algorithms. In fact, for all the different *matching rates* considered, MCL-AG can be found to be able to identify more protein complexes for both *Krogan 2006* and *DIP Scere.* Hence, if we prefer to have a protein-identification algorithm that can identify more protein complexes for verification and validation while maintaining a relatively high matching rate, MCL-AG is preferred.

Regarding the performance of CAP-AG in terms of *mr*, it is observed from Tables 16-18 that CAP-AG has a comparable performance when compared with MCL-AG for yeast PPINs. In particular, for *Krogan 2006*, CAP-AG is the best algorithm while MCL-AG ranks as the second best algorithm; for *DIP Scere*, MCL-AG is the best algorithm while CAP-AG ranks as the second best algorithm. But for the human PPIN *DIP-Hsapi*, CAP-AG does not perform as well as MCL-AG. Considering the difference between the density of yeast PPIN and that of Human PPIN, CAP-AG may not obtain a satisfactory performance in a network that is much sparse.

For *DIP Hsapi*, RNSC, however, performs better than both of MCL-AG and CAP-AG when $\partial \geq 0.9$. This is because RNSC is developed to find an optimal partitioning of a graph and the native cost function it uses as a criterion of partitioning costs allows it to be able to better identify two denser regions connected by a sparse region when a PPI network is sparse. For MCL-AG and CAP-AG, these regions will probably be identified as one whole region.

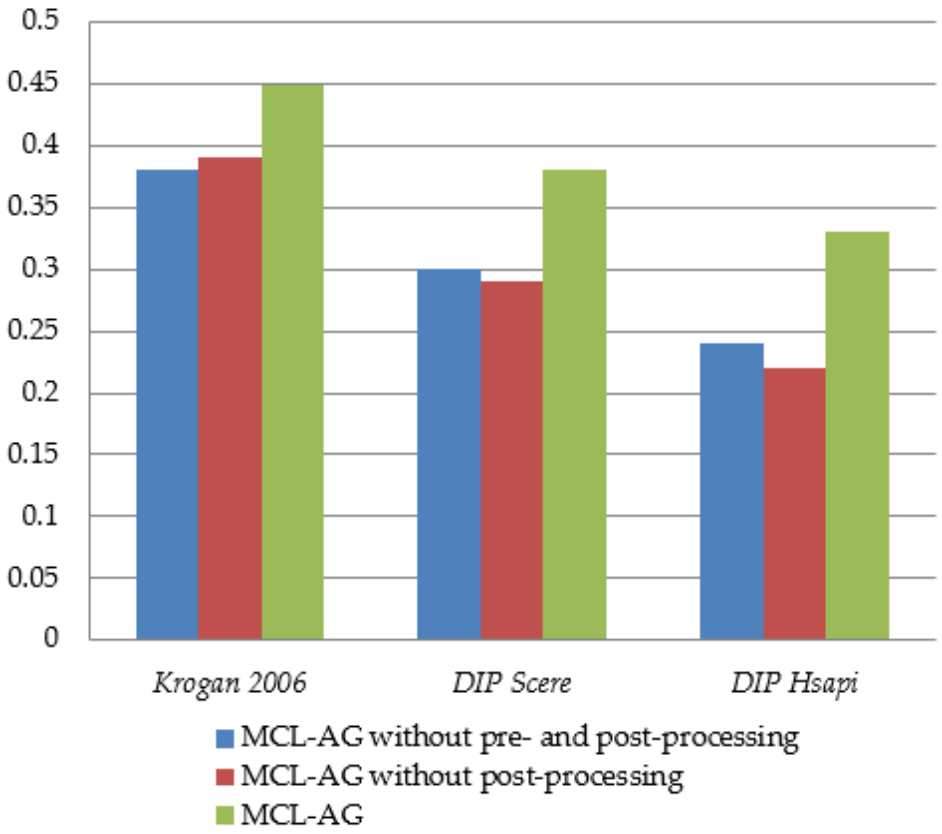

Figure 14. The *f*-measure results of MCL-AG without pre- and post-processing, MCL-AG without post-processing and MCL-AG.

In general, as a protein complex identification algorithm, both of MCL-AG and CAP-AG have a promising performance. They are better than MCL or other algorithms from the perspectives of *matching rates* and *f*-measure.

B. Performance Analysis of MCL-AG

1) Impact of Each Step of MCL-AG on Its Performance

The first step of MCL-AG can be considered as the pre-processing step for MCL as this step has to be performed before MCL to find weights for the adjacency matrix needed for MCL to perform its tasks. The weights allow MCL to take into consideration the protein attributes or GO information. The last step of MCL-AG has to be performed after MCL completes its tasks and can be considered as a post-processing step for MCL to partition the graph clusters into protein complexes to ensure that the proteins have significant association relationship with each other. To

investigate into the impact that each of these steps of MCL-AG may have on its performance so that we can understand which of the steps of MCL-AG contributes the most significantly to its better performance, we conducted several experiments.

The first test we performed was to skip the Step 1(i.e., the pre-processing steps) but, by skipping Step 1, it should be noted that Step 3 (i.e., the post-processing step) was also not applicable as there was no weighted PPIN for it to use for cluster partitioning. Hence, by skipping Steps 1 and 2, we are basically using MCL alone without the pre- and post-processing. For the second test, we kept the pre-processing step but skipped Step 3.

Based on the *f*-measure results as shown in Figure 14, we find that without either the pre- or post-processing steps in MCL-AG, its performance is negatively impacted as in either case, the GO information in the PPI network is not taken into full consideration. There does not seem to be too much difference in performance between skipping the pre-processing steps and skipping the post-processing step.

2) Effect of μ on Performance of MCL-AG

The parameter μ is introduced in the final step of MCL-AG so that users can decide how strong the proteins should associate with each other for them to form a complex. To find out how the setting of μ may have on the performance of MCL-AG, we conducted sensitivity tests using all three PPI network data sets of *Krogan 2006*, *DIP Scere* and *DIP Hsapi*.

In our tests, we recorded the *f*-measure scores as well as the average sizes of protein complexes detected when μ is set to 0, 0.1, 0.2, …, 1.0, and the results are shown in Figure 16 and Figure 15 respectively. For the purpose of the sensitivity tests, the matching rate used to mark identified protein complexes as "matched" was set to 0.2.

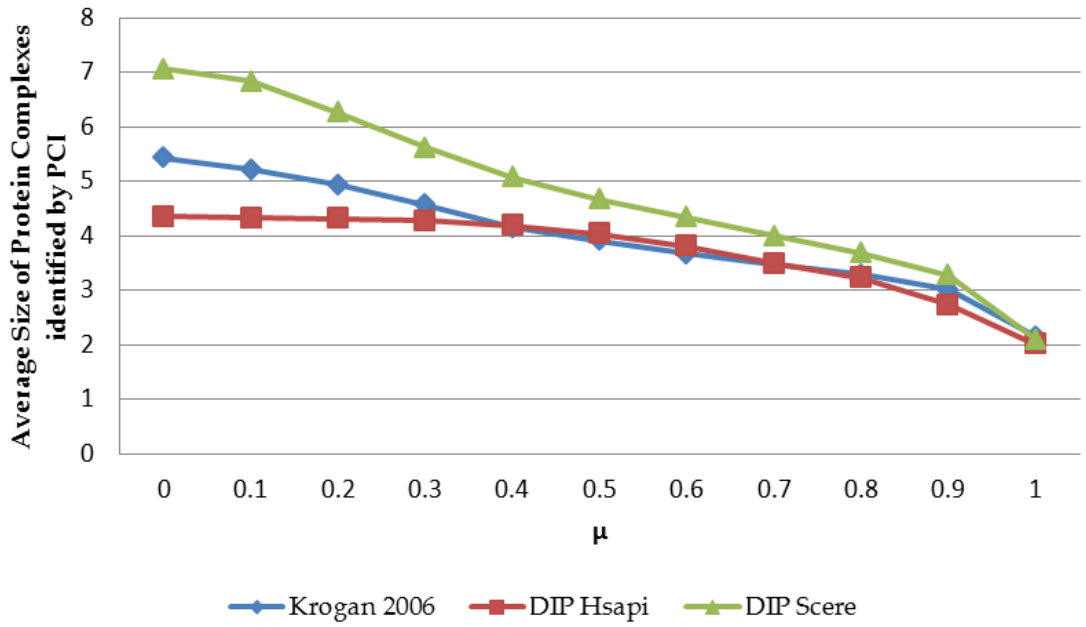

Figure 16. The average sizes of protein complexes identified by MCL-AG with Different Values of μ in the PPI networks of Krogan 2006, DIP Hsapi and DIP Scere.

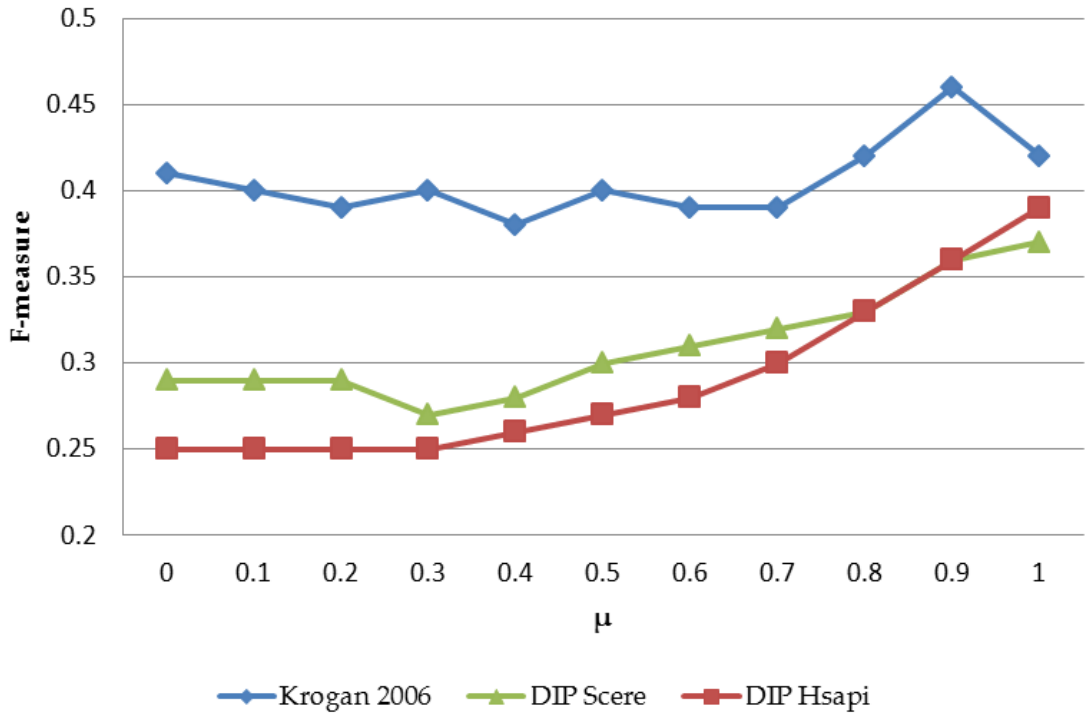

Figure 15. The *f*-measure results of MCL-AG with Different Values of μ in the PPI networks of Krogan 2006, DIP Scere and DIP Hsapi.

From Figure 16, it is noted that the *f*-measure curves are generally upward sloping along with the increase of μ. In particular, the *f*-measure scores are insensitive to the

low values of μ as the changes in the values of *f*-measure are rather small when μ is within [0, 0.4]. A steady improvement can be observed when μ is given a value within [0.5, 1.0]. The reason why the *f*-measure results seem to be rather insensitive to small μ is due to the fact that the difference between the highest and the lowest weight in a subgraph found after Step 2 in MCL-AG is used as a threshold for the setting of μ and if we adopt a μ smaller than this threshold, there will not be much of an effect on performance.

With regard to the impact of μ on the average size of the protein complexes identified by MCL-AG, we note that all the curves of average size exhibit a downward trend (see Figure 15). This means that the larger the value of μ is, the smaller the average size of identified protein complexes is. However, if the value of μ is set to a large value of, say, 0.9 or 1.0, it is possible for a cluster discovered after Step 2 to be "over-partitioned" and as a result, most protein complexes identified will be very small. As a trade-off, μ is best selected to balance between having a better *f*-measure performance and a reasonable large average size of identified protein complexes. Hence, we recommend that μ would be set between 0.65 and 0.85. Values set within this range will ensure that MCL-AG performs reasonably well. In our experiments, we set μ to be 0.8.

3) Protein Complexes Identified – Biological Significance

Other than the objective statistics in terms of the *f*-measure or the number of known protein complexes discovered, we have also looked into the details of the protein complexes identified by the MCL-AG to see if there is anything biologically significant in them that can help us better understand the protein complexes.

To facilitate our investigation, we performed a functional enrichment analysis using the tool GO::TermFinder [10]. The GO::TermFinder is provided by SGD [20] as a web service and it can be used to search for significant shared GO terms in the proteins in a protein complex identified by MCL-AG based on p-value tests. For our analysis,

the GO terms with p-values smaller than or equal to some significant threshold of 0.05 in the proteins of a protein complex are identified. The enrichment of these GO terms is indicative of their significance as the co-occurrences of such GO terms in protein complexes are not likely to occur by chance [93]. In Figure 17, we list among the protein complexes identified by MCL-AG in *Krogan 2006* and *DIP Scere* some of those that contain at least one GO term with *p*-value ≤ 0.05 in each GO category. These protein complexes are not recorded in MIPS/CYGD or CYC2008 but as can be seen from the significant p-values, they are all likely candidates of real protein complexes because of the statistical significance indicated by the functional enrichment analysis. Careful analysis of all protein complexes identified by MCL-AG reveals that, for *Krogan 2006*, 770 out of the 1210 protein complexes identified, and for *DIP Scere*, 1092 out of the 1823 protein complexes identified by MCL-AG are considered significant with corrected p-value ≤ 0.05. As the current version of the GO::TermFinder only supports p-value tests for the proteins of Saccharomyces Cerevisiae, we only performed such analysis with *Krogan 2006* and *DIP Scere*.

In addition to the functional enrichment analysis, we also selected some typical protein complexes identified by MCL-AG to analyze from the aspects of both the structure and the GO information.

One of the protein complexes identified by MCL-AG in *Krogan 2006* matches with the Anaphasa promoting complex (APC with MIPS/CYGD ID 60). It is an ubiquitin ligase that has essential functions in and outside the eukaryotic cell cycle [68] and its topology is presented in Figure 17. MCL-AG successfully identified 9 of the 11 proteins that constitute the APC. From Figure 17, it should be noted that the protein with ID Q12157 is not connected with any other proteins in the APC complex and this is also the reason why MCL-AG could not identify it as a part of the APC.

To understand how important the consideration of protein attributes has helped identify APC, we compare it with the APC identified by MCL. For MCL, the APC that it identified had 12 proteins altogether. Of these 12 proteins, nine of them are the

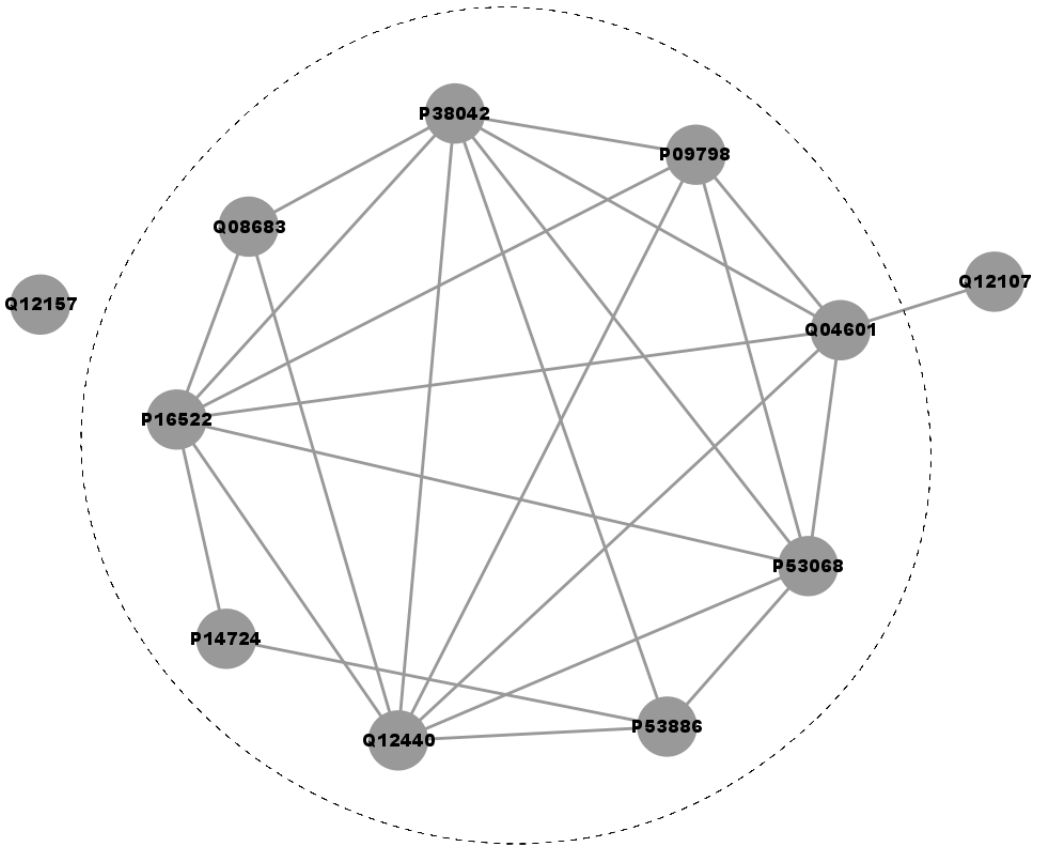

Figure 17. The MAPPIN graph of the Anaphase Promoting Complex (APC) where the part circled by dashed line is the cluster identified by MCL-AG from *Krogan 2006*.

same as the known APC. The remaining three proteins with IDs of Q08646, P40577 and P40008 are not part of the known APC as found in MIPS/CYGD. The use of protein attributes has, therefore, helped filter out these three proteins. The weights of the edges between them and the other nine proteins were too small to allow them to be included in the APC as identified by the MCL-AG.

When considering the attribute, *molecular functions*, of the proteins, it is observed that, in addition to being able to identify all proteins that perform the same function of ubiquitin-protein ligase activity (GO:0004842), MCL-AG is also able to identify the protein with ID P53068 which performs a different function. In other words, if MCL-AG only assumes function homogeneity when identifying protein complexes, it will miss the protein P53068 as part of APC.

The protein P53068 interacts with a total of six other proteins in the APC. To find out how the attributes of these interacting proteins may relate to each other, let us consider the protein with ID Q04061. In Table 19, if an attribute value of P53068 has a significant association relationship with an attribute value of Q04061, it is indicated with “Y". From these results, we notice that there exist many significant association relationships between the *molecular functions* of the two integrating proteins that are captured by the DOA measure. Furthermore we found that there was a high ratio of associating attribute values from the possible combinations of values of other attributes. Hence, with the consideration of other attributes, i.e., *biological process* and *cellular component*, as well as the correlation between attribute values, MCL-AG is capable of catching such exceptional proteins in terms of the function homogeneity.

Table 19. The Association between Some of the Protein Attribute Values in Q04601 and P53068 respectively in *Krogan 2006*

			Q04601						
			Λ_p			Λ_f	Λ_c		
			GO:0007049	GO:0031145	…	GO:0004842	GO:0005634	GO:0005737	…
P53068	Λ_p	GO:0007049	Y*	Y		Y	N	N	
		GO:0008054	Y	Y		Y	Y	N	
		…	…						
	Λ_f	GO:0030234	N*	Y		Y	Y	N	
		GO:0090302	Y	Y		Y	N	N	
	Λ_c	GO:0005634	N	Y		N	Y	N	
		GO:0005737	N	Y		Y	N	Y	
		……	……						

* Y = the two corresponding attribute values have significant association, N= they do not

C. Performance Analysis of CAP-AG

We have also looked into the details of protein complexes identified by the proposed approach to see whether the consideration of attribute preferences can actually facilitate the prediction process.

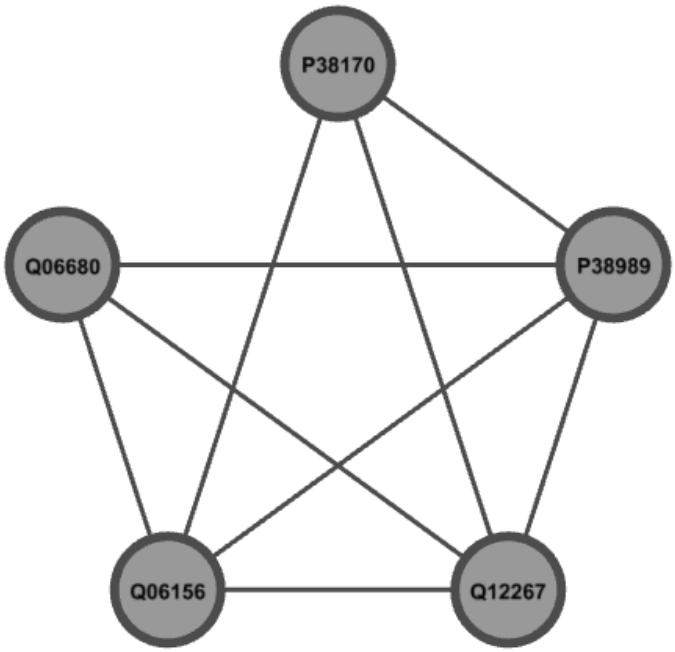

Figure 18. The network topology of Nuclear Condensin Complex identified by CAP-AG from *Krogan 2006*.

One of the protein complexes identified by CAP-AG in *Krogan 2006* is exactly matched with a known protein complex, i.e., Nuclear Condensin Complex (NCC), whose topology network is presented in Figure 18. The protein attributes of NCC are given in Table 5. From Figure 18, it is observed that proteins in NCC are densely connected and hence the optimization problem of (35) is proved to have the ability of retaining proteins that are densely connected to compose protein complexes. Regarding the protein attribute of proteins in NCC, we note that the three proteins, i.e., Q06156, Q06680 and P38170, are different from the other two proteins of P38989 and Q12267 in terms of the attribute of *molecular function*, as these two sets of proteins have no value in common. In other words, the attributes of *biological process* and *cellular component* are more important w.r.t. the formation of NCC. However, approaches that make use of similarity to process the attribute information could probably miss this protein complex because of the low similarity scores in protein attributes. With the introduction of attribute preferences, CAP-AG has overcome this problem. In particular, when identifying NCC, the proposed approach assigned high preference values to the attributes of *biological process* and *cellular component* and

low preference value to the attribute of molecular function when computing the preference vectors of proteins in NCC. Therefore, NCC has been identified by CAP-AG.

Table 20. The protein attributes of proteins in Nuclear Condensin Complex

Protein	Λ_p	Λ_f	Λ_c
P38989	GO:0000070, GO:0006281, GO:0006310 GO:0007049, GO:0007062, GO:0007067 GO:0007076, GO:0030261, GO:0051276 GO:0051301, GO:0070058	GO:0000166, GO:0000217 GO:0003680 GO:0003690 GO:0005524 GO:0016887	GO:0005634, GO:0005694 GO:0005737, GO:0005739
Q12267	GO:0000070, GO:0006281, GO:0006310 GO:0007049, GO:0007062, GO:0007067 GO:0007076, GO:0030261, GO:0051276 GO:0051301, GO:0070058	GO:0000166 GO:0003682 GO:0003682 GO:0005524 GO:0016887	GO:0005634, GO:0005694 GO:0005737
Q06156	GO:0000070, GO:0007049, GO:0007067 GO:0007076, GO:0010032, GO:0030261 GO:0030466, GO:0043007, GO:0051301 GO:0070058	GO:0003674	GO:0005634, GO:0005694 GO:0005730
Q06680	GO:0000070, GO:0007049, GO:0007067 GO:0007076, GO:0010032, GO:0030261 GO:0051301, GO:0070058, GO:0070550	GO:0003674	GO:0005634, GO:0005694 GO:0005737
P38170	GO:0000070, GO:0007049, GO:0007067 GO:0007076, GO:0030261, GO:0051301 GO:0070058	GO:0003674	GO:0005634, GO:0005694 GO:0005737

5.5 Conclusion

In this chapter, we have studied the problem of AG clustering without knowing the number of clusters and propose two approaches MCL-AG and CAP-AG to tackle this problem from different perspectives. In particular, MCL-AG makes use of DOA measure to indicate how likely two vertices are grouped in the same cluster and then identify the clusters through a markov clustering process; for CAP-AG, attributes are considered separately so as to disclose the dependency between them and the resultant clusters. To evaluate the performance of MCL-AG and CAP-AG, we apply them to the application of identifying protein complexes from PPINs. Experiment results show both of MCL-AG and CAP-AG are capable of identifying protein complexes

efficiently. Furthermore, MCL-AG and CAP-AG have a comparable performance for relatively dense PPINs, but for sparse PPINs, MCL-AG obtains a better performance.

Chapter 6 Conclusion and Future Work

6.1 Conclusion

In this thesis, we have studied the problem of discovering associative patterns from complex networks with attribute information available. Based on these associative patterns, we develop several approaches to tackle some of AG applications including link analysis and graph clustering. Motivated by the idea that existing approaches proposed for AG applications normally consider the information of graph topology and that of vertex attributes separately, we intends to discover patterns that are able to discover the associations between graph topology and vertex attributes and hence the two kinds of information in AGs can be jointly considered. Regarding the presence of these patterns in AGs, since it is intuitive to believe that pairwise attribute values that frequently co-occur in adjacent vertices, to some extent, represent such kind of patterns, we propose an innovative algorithm to statistically discover these pairwise attribute values and define them as the associative patterns that consider the topology information and the attribute information simultaneously. In addition to the discovery of associative patterns, we also define a DOA measure to weight the association between vertices according to the amount and the significances of associative patterns found in their attributes.

To evaluate the significance of associative patterns, we apply them to some of AG applications and propose different approaches to solve specific problems. The extensive experimental results show that the consideration of these patterns can significantly improve the performance.

For the application of link analysis, we are particularly interested in the prediction of PPIs from PPINs based on sequence information. To solve it, we propose an approach VLASPD that makes use of variable-length sequence segments to compose the

patterns for the prediction task. The promising performance of VLASPD shows that variable-length sequence segments can facilitate the prediction of PPIs as more information is considered when compared with the use of fixed-length sequence segments.

When applying the associative patterns to the application of graph clustering, we propose FC-AG to identify clusters from AGs in a fuzzy manner. The experimental results of FC-AG on the problems of document classification and social community detection show that FC-AG outperforms the state-of-the-art approaches and also FC-AG has a linear running time over the increase of network size. In addition to FC-AG, two unsupervised clustering approaches MCL-AG and CAP-AG have been developed for the clustering problem without knowing the number of clusters. MCL-AG performs the task through a markov clustering process based on associative patterns while CAP-AG identified clusters from AGs by considering the attribute preferences and link structure simultaneously. To evaluate the performances of MCL-AG and CAP-AG, we apply them to solve the problem of identifying protein complexes from PPINs. The experimental results show that 1) both of MCL-AG and CAP-AG can identify the protein complexes efficiently, and 2) CAP-AG is more sensible to the density of network while MCL-AG is more stable.

6.2 Future Works

The future works will be unfolded from three aspects as below.

1) In the problem of PPI prediction, we compose a rule set based on the patterns discovered by VLASPD. However, through the experimental results, we find that the number of rules is relatively huge and hence we have to prune the rule set by only keeping those with best performance. Furthermore, it is seen that some templates of rules can be found if we consider wildcards and gaps to formulate rules. Therefore, if we have a way to consolidate rules by finding

out the common templates, it is possible to have a more accurate rule set for the prediction task.

2) The concept of big data has been being a quite popular topic in last decade and obviously there are also large AGs that contains more than one million nodes for analysis. However, existing approaches proposed for the applications of AGs seldom consider the case of large AGs and therefore these approaches could become slow when applied to AGs graphs. Therefore, like what we have done to FC-AG, we would like to modify the other approaches proposed in this thesis so that they can be executed in the environment of distributed computing in order to solve the applications of large AGs.

3) Although there are a number of applications of AGs, we have only studied two of them in this thesis. In future, we would like to extend our research by involving more applications of AGs, such as peer influence in social networks, the network traffic classification and the identification of essential proteins from PPINs.

References

[1] Adamcsek, B., Palla, G., Farkas, I. J., Derényi, I., & Vicsek, T. (2006). CFinder: Locating cliques and overlapping modules in biological networks. *Bioinformatics*, *22*(8), 1021-1023.

[2] Aggarwal, C. C. (2011). An introduction to social network data analytics. Springer US.

[3] Agrawal, R., and Srikant, R. (1994). Fast algorithms for mining association rules. In *Proceedings of 20th International Conference on very Large Data Bases* (pp. 487-499).

[4] Altaf-Ul-Amin, M., Shinbo, Y., Mihara, K., Kurokawa, K., & Kanaya, S. (2006). Development and implementation of an algorithm for detection of protein complexes in large interaction networks. *BMC Bioinformatics*, *7*(1), 207.

[5] Ashburner, M., Ball, C. A., Blake, J. A., Botstein, D., Butler, H., Cherry, J. M., et al. (2000). Gene ontology: Tool for the unification of biology. *Nature Genetics*, *25*(1), 25-29.

[6] Bader, G. D., & Hogue, C. W. (2003). An automated method for finding molecular complexes in large protein interaction networks. *BMC Bioinformatics*, *4*(1), 2.

[7] Batuwita, R., & Palade, V. (2013). Class imbalance learning methods for support vector machines. *Imbalanced Learning: Foundations, Algorithms, and Applications*. John Wiley & Sons.

[8] Ben-Hur, A., & Noble, W. S. (2005). Kernel methods for predicting protein-protein interactions. *Bioinformatics*, *21*(suppl 1), i38-46.

[9] Bock, J. R., & Gough, D. A. (2001). Predicting protein--protein interactions from primary structure. *Bioinformatics*, *17*(5), 455-460.

[10] Boyle, E. I., Weng, S., Gollub, J., Jin, H., Botstein, D., Cherry, J. M., et al. (2004). GO::TermFinder—open source software for accessing gene ontology information and finding significantly enriched gene ontology terms associated with a list of genes. *Bioinformatics*, *20*(18), 3710-3715.

[11] Brandes, U., et al. (2008). On modularity clustering. *IEEE Transactions on Knowledge and Data Engineering*, *20*(2), 172-188.

[12] Bron, C., & Kerbosch, J. (1973). Algorithm 457: Finding all cliques of an undirected graph. *Communications of the ACM*, *16*(9), 575-577.

[13] Brohee, S., & van Helden, J. (2006). Evaluation of clustering algorithms for protein-protein interaction networks. *BMC Bioinformatics*, *7*(1), 488.

[14] Camon, E., M., Camon, E., Magrane, M., Barrell, D., Lee, V., et al. (2003). The gene ontology annotation (GOA) database: Sharing knowledge in uniprot with gene ontology. *Nucleic Acids Research*, *32*(suppl 1), D262-D266.

[15] Chan, K. C. C., Wong, A. K. C., & Chiu, D. K. Y. (1994). Learning sequential patterns for probabilistic inductive prediction. *IEEE Transactions on Systems, Man and Cybernetics*, *24*(10), 1532-1547.

[16] Chang, C., & Lin, C. (2011). LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, *2*(3), 27.

[17] Chen, J., & Saad, Y. (2012). Dense subgraph extraction with application to community detection. *IEEE Transactions on Knowledge and Data Engineering*, *24*(7), 1216-1230.

[18] Chen, X. W., & Liu, M. (2005). Prediction of protein-protein interactions using random decision forest framework. *Bioinformatics*, *21*(24), 4394-4400.

[19] Cheng, H., Zhou, Y., & Yu, J. X. (2011). Clustering large attributed graphs: A balance between structural and attribute similarities. *ACM Transactions on Knowledge Discovery from Data*, *5*(2), 12.

[20] Cherry, J. M., Adler, C., Ball, C., Chervitz, S. A., Dwight, S. S., Hester, E. T., et al. (1998). SGD: Saccharomyces genome database. *Nucleic Acids Research*, *26*(1), 73-79.

[21] Chien, C. et al. (1991). The two-hybrid system: A method to identify and clone genes for proteins that interact with a protein of interest. *Proceedings of the National Academy of Sciences*, *88*(21), 9578-9582.

[22] Dandekar, T., Snel, B., Huynen, M., & Bork, P. (1998). Conservation of gene order: A fingerprint of proteins that physically interact. *Trends in Biochemical Sciences*, *23*(9), 324-328.

[23] Deisboeck, T., & Kresh, J. Y. (2006). Complex systems science in BioMedicine (1st ed.), Springer.

[24] Deng, M., Mehta, S., Sun, F., & Chen, T. (2002). Inferring domain-domain interactions from protein-protein interactions. *Genome Research*, *12*(10), 1540-1548.

[25] Dhillon, I. S. (2001). Co-clustering documents and words using bipartite spectral graph partitioning. In *Proceedings of the 2001 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 269-274).

[26] Ding, X., Wang, W., Peng, X., & Wang, J. (2012). Mining protein complexes from PPI networks using the minimum vertex cut. *Tsinghua Science and Technology*, *17*(6), 674-681.

[27] Dongen, S. M. v. (2000). *Graph clustering by flow simulation*. PhD Thesis. University of Utrecht: the Netherlands.

[28] Due, P., Holstein, B., Lund, R., Modvig, J., & Avlund, K. (1999). Social relations: Network, support and relational strain. *Social Science & Medicine*, *48*(5), 661-673.

[29] Enright, A. J., Iliopoulos, I., Kyrpides, N. C., & Ouzounis, C. A. (1999). Protein interaction maps for complete genomes based on gene fusion events. *Nature*, *402*(6757), 86-90.

[30] Ester, M., Kriegel, H., Sander, J., & Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the 1996 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 226-231).

[31] Flake, G. W., Tarjan, R. E., and Tsioutsiouliklis, K. (2004). Graph clustering and minimum cut trees. *Internet Mathematics*, *1*(4), 385-408.

[32] Gavin, A. C., Bosche, M., Krause, R., Grandi, P., et al. (2002). Functional organization of the yeast proteome by systematic analysis of protein complexes. *Nature*, *415*(6868), 141-147.

[33] Girvan, M., and Newman, M. E. (2002). Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, *99*(12), 7821-7826.

[34] Görke, R., Dehne, F., Gavrilova, M., Sack, J., & Tóth, C. D. (2009). Dynamic graph clustering using minimum-cut trees. Springer Berlin Heidelberg.

[35] Güldener, U., Münsterkötter, M., Kastenmüller, G., Strack, N., van Helden, J., Lemer, C., et al. (2005). CYGD: The comprehensive yeast genome database. *Nucleic Acids Research*, *33*(suppl 1), D364-D368.

[36] Günnemann, S., Boden, B., and Seidl, T. (2011). DB-CSC: a density-based approach for subspace clustering in graphs with feature vectors. *Machine Learning and Knowledge Discovery in Databases*. Springer Berlin Heidelberg.

[37] Haberman, S. J. (1973). The analysis of residuals in cross-classified tables. *Biometrics*, *29*(1), 205-220.

[38] Han, J., Kamber, M., & Pei, J. (2006). Data mining: Concepts and techniques. Morgan kaufmann.

[39] Ho, Y., Gruhler, A., Heilbut, A., Bader, G. D., Moore, L., Adams, S., et al. (2002). Systematic identification of protein complexes in saccharomyces cerevisiae by mass spectrometry. *Nature*, *415*(6868), 180.

[40] Hu, A. L., & Chan, K. (2013). Utilizing both topological and attribute information for protein complex identification in PPI networks. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, *10*(3), 780-792.

[41] Ito, T., Chiba, T., Ozawa, R., Yoshida, M., Hattori, M., & Sakaki, Y. (2001). A comprehensive two-hybrid analysis to explore the yeast protein interactome. *Proceedings of the National Academy of Sciences*, *98*(8), 4569-4574.

[42] Jansen, R., Yu, H., Greenbaum, D., Kluger, Y., Krogan, N. J., Chung, S., et al. (2003). A bayesian networks approach for predicting protein-protein interactions from genomic data. *Science*, *302*(5644), 449-453.

[43] Kanaan, S. P., Huang, C., Wuchty, S., Chen, D. Z., & Izaguirre, J. A. (2009). Inferring protein–protein interactions from multiple protein domain combinations. *Computational systems biology*. Springer.

[44] Ketkar, N. S., Holder, L. B., & Cook, D. J. (2005). Subdue: Compression-based frequent pattern discovery in graph data. In *Proceedings of the 2005 International Workshop on Open Source Data Mining: Frequent Pattern Mining Implementations* (pp. 71-76).

[45] King, A. D., Pržulj, N., & Jurisica, I. (2004). Protein complex prediction via cost-based clustering. *Bioinformatics*, *20*(17), 3013-3020.

[46] Krogan, N. J., Cagney, G., Yu, H., Zhong, G., Guo, X., Ignatchenko, A., et al. (2006). Global landscape of protein complexes in the yeast saccharomyces cerevisiae. *Nature*, *440*(7084), 637-643.

[47] Kuramochi, M., & Karypis, G. (2001). Frequent subgraph discovery. Paper presented at the Data Mining. In *Proceedings of the 2001 IEEE International Conference on Data Mining* (pp. 313-320).

[48] Lam, W. W. M., & Chan, K. C. C. (2012). Discovering functional interdependence relationship in PPI networks for protein complex identification. *IEEE Transactions on Biomedical Engineering*, *59*(4), 899-908.

[49] Li, M., Chen, J., Wang, J., Hu, B., & Chen, G. (2008). Modifying the DPClus algorithm for identifying protein complexes based on new topological structures. *BMC Bioinformatics*, *9*(1), 398.

[50] Li, M., Wu, X., Wang, J., & Pan, Y. (2012). Towards the identification of protein complexes and functional modules by integrating PPI network and gene expression data. *BMC Bioinformatics*, *13*(1), 109.

[51] Li, X., Wu, M., Kwoh, C., & Ng, S. (2010). Computational approaches for detecting protein complexes from protein interaction networks: A survey. *BMC Genomics*, *11*(suppl 1), S3.

[52] Liu, G., Wong, L., & Chua, H. N. (2009). Complex discovery from weighted PPI networks. *Bioinformatics*, *25*(15), 1891-1897.

[53] Luxburg, U. (2007). A tutorial on spectral clustering. *Statistics and Computing*, *17*(4), 395-416.

[54] Mahdavi, M. A., & Lin, Y. (2007). Prediction of protein-protein interactions using protein signature profiling. *Genomics, Proteomics & Bioinformatics*, *5*(3), 177-186.

[55] Marcotte, E. M., Pellegrini, M., Ng, H. L., Rice, D. W., Yeates, T. O., & Eisenberg, D. (1999). Detecting protein function and protein-protein interactions from genome sequences. *Science*, *285*(5428), 751-753.

[56] Marcotte, E. M., Pellegrini, M., Thompson, M. J., Yeates, T. O., & Eisenberg, D. (1999). A combined algorithm for genome-wide prediction of protein function. *Nature*, *402*(6757), 83-86.

[57] Martin, S., Roe, D., & Faulon, J. L. (2005). Predicting protein-protein interactions using signature products. *Bioinformatics*, *21*(2), 218-226.

[58] Mei, J., & Chen, L. (2012). A fuzzy approach for multitype relational data clustering. *IEEE Transactions on Fuzzy Systems*, *20*(2), 358-371.

[59] Nepusz, T., Petróczi, A., Négyessy, L., & Bazsó, F. (2008). Fuzzy communities and the concept of bridgeness in complex networks. *Physical Review E*, *77*(1), 016107.

[60] Newman, M. E. J. (2006). Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, *103*(23), 8577-8582.

[61] Owen, S., Anil, R., Dunning, T., & Friedman, E. (2011). Mahout in action. Manning.

[62] Park, Y. (2009). Critical assessment of sequence-based protein-protein interaction prediction methods that do not require homologous protein sequences. *BMC Bioinformatics*, *10*(1), 419.

[63] Park, Y., and Marcotte, E. M. (2011). Revisiting the negative example sampling problem for predicting protein–protein interactions. *Bioinformatics*, *27*(21), 3024-3028.

[64] Parrish, J. R. et al. (2006). Yeast two-hybrid contributions to interactome mapping. *Current Opinion in Biotechnology*, *17*(4), 387-393.

[65] Pawson, T., & Nash, P. (2003). Assembly of cell regulatory systems through protein interaction domains. *Science Signaling*, *300*(5618), 445.

[66] Pazos, F., & Valencia, A. (2001). Similarity of phylogenetic trees as indicator of protein-protein interaction. *Protein Engineering*, *14*(9), 609-614.

[67] Pellegrini, M., Marcotte, E. M., Thompson, M. J., Eisenberg, D., & Yeates, T. O. (1999). Assigning protein functions by comparative genome analysis:

Protein phylogenetic profiles. *Proceedings of the National Academy of Sciences*, *96*(8), 4285-4288.

[68] Peters, J. (2006). The anaphase promoting complex/cyclosome: A machine designed to destroy. *Nature Reviews Molecular Cell Biology*, *7*, 644-656.

[69] Pitre, S., Dehne, F., Chan, A., Cheetham, J., Duong, A., Emili, A., et al. (2006). PIPE: A protein-protein interaction prediction engine based on the re-occurring short polypeptide sequences between known interacting protein pairs. *BMC Bioinformatics*, *7*, 365.

[70] Pitre, S., Alamgir, M., Green, J. R., Dumontier, M., Dehne, F., & Golshani, A. (2008). Computational methods for predicting protein–protein interactions. *Protein–Protein interaction*. Springer.

[71] Prasad, T. K., Goel, R., Kandasamy, K., Keerthikumar, S., Kumar, S., Mathivanan, S., et al. (2009). Human protein reference database—2009 update. *Nucleic Acids Research*, *37*(suppl 1), D767-D772.

[72] Pu, S., Wong, J., Turner, B., Cho, E., & Wodak, S. J. (2009). Up-to-date catalogues of yeast protein complexes. *Nucleic. Acids. Res.*, *37*(3), 825-831.

[73] Rardin, R. L. (1998). Optimization in operations research. Prentice Hall: New Jersey.

[74] Rives, A. W., and Galitski, T. (2003). Modular organization of cellular networks. *Proceedings of the National Academy of Sciences*, *100*(3), 1128-1133.

[75] Ruan, Y., Fuhry, D., and Parthasarathy, S. (2013). Efficient community detection in large networks using content and links. In *Proceedings of the 2013 International Conference on World Wide Web*.

[76] Ruepp, A., Brauner, B., Dunger-Kaltenbach, I., Frishman, G., Montrone, C., Stransky, M., et al. (2008). CORUM: The comprehensive resource of mammalian protein complexes. *Nucleic Acids Research*, *36*(suppl 1), D646-D650.

[77] Schaeffer, S. E. (2007). Graph clustering. *Computer Science Review*, *1*(1), 27-64.

[78] Sen, P., Namata, G. M., Bilgic, M., Getoor, L., Gallagher, B., & Eliassi-Rad, T. (2008). Collective classification in network data. *AI Magazine*, *29*, 93-106.

[79] Shen, J., Zhang, J., Luo, X., Zhu, W., Yu, K., Chen, K., et al. (2007). Predicting protein–protein interactions based only on sequences information. *Proceedings of the National Academy of Sciences*, *104*(11), 4337-4341.

[80] Spirin, V., & Mirny, L. A. (2003). Protein complexes and functional modules in molecular networks. *Proceedings of the National Academy of Sciences*, *100*(21), 12123-12128.

[81] Stark, C., Breitkreutz, B., Reguly, T., Boucher, L., Breitkreutz, A., & Tyers, M. (2006). BioGRID: A general repository for interaction datasets. *Nucleic Acids Research, 34*(Suppl 1), D535-D539.

[82] Stix, V. (2004). Finding all maximal cliques in dynamic graphs. *Computational Optimization and Applications*, *27*(2), 173-186.

[83] Studholme, C., Hill, D. L., & Hawkes, D. J. (1999). An overlap invariant entropy measure of 3D medical image alignment. *Pattern Recognition*, *32*(1), 71-86.

[84] Sun, Y., Han, J., Gao, J., & Yu, Y. (2009). iTopicModel: Information network-integrated topic modeling. In *Proceedings of the 2009 IEEE International Conference on Data Mining* (pp. 493-502).

[85] Templin, M. F., et al. (2003). Protein microarrays: Promising tools for proteomic research. *Proteomics*, *3*(11), 2155-2166。

[86] Toivonen, R., Kovanen, L., Kivelä, M., Onnela, J. P., Saramäki, J., & Kaski, K. (2009). A comparative study of social network models: Network evolution models and nodal attribute models. *Social Networks*, *31*(4), 240-254.

[87] Tong, A. H. Y., Drees, B., Nardelli, G., Bader, G. D., Brannetti, B., Castagnoli, L., et al. (2002). A combined experimental and computational strategy to define protein interaction networks for peptide recognition modules. *Science*, *295*(5553), 321-324.

[88] Tsoka, S., & Ouzounis, C. A. (2000). Prediction of protein interactions: Metabolic enzymes are frequently involved in gene fusion. *Nature Genetics*, *26*(2), 141-142.

[89] Trinkle-Mulcahy, L., et al. (2008). Identifying specific protein interaction partners using quantitative mass spectrometry and bead proteomes. *The Journal of Cell Biology*, *183*(2), 223-239.

[90] Van Dongen, S. (2000). A cluster algorithm for graphs. *Report-Information Systems*, *10*, 1-40.

[91] Wang, Y., & Wong, A. K. C. (2003). From association to classification: Inference using weight of evidence. *IEEE Transactions on Knowledge and Data Engineering*, *15*(3), 764-767.

[92] Wheeler, D. L., Barrett, T., Benson, D. A., Bryant, S. H., Canese, K., Chetvernin, V., et al. (2007). Database resources of the national center for biotechnology information. *Nucleic Acids Research*, *35*(suppl 1), D5-D12.

[93] Wu, M., Li, X., Kwoh, C., & Ng, S. (2009). A core-attachment based method to detect protein complexes in PPI networks. *BMC Bioinformatics*, *10*(1), 169.

[94] Xenarios, I., Salwínski, L., Duan, X. J., Higney, P., Kim, S., & Eisenberg, D. (2002). DIP, the database of interacting proteins: A research tool for studying cellular networks of protein interactions. *Nucleic Acids Research*, *30*(1), 303-305.

[95] Xu, Z., Ke, Y., Wang, Y., Cheng, H., & Cheng, J. (2012). A model-based approach to attributed graph clustering. In *Proceedings of the 2012 ACM SGIDMOD International Conference on Management of Data* (pp. 505-516).

[96] Yan X., & Han, J. (2002). gSpan: Graph-based substructure pattern mining. In *Proceedings of the 2002 IEEE International Conference on Data Mining* (pp. 721-724).

[97] Yang, J., McAuley, J., & Leskovec, J. (2013). Community Detection in Networks with Node Attributes. In *Proceedings of the 2013 IEEE International Conference on Data Mining*.

[98] Yang, T., Jin, R., Chi, Y., & Zhu, S. (2009). Combining link and content for community detection: a discriminative approach. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

[99] You, Z. H. et al. (2010). Using manifold embedding for assessing and predicting protein interactions from high-throughput experimental data. *Bioinformatics*, *26*(21), 2744-2751.

[100] Yu, J. et al. (2010). Simple sequence-based kernels do not predict protein–protein interactions. *Bioinformatics*, *26*(20), 2610-2614.

[101] Zahiri, J., Yaghoubi, O., Mohammad-Noori, M., Ebrahimpour, R., & Masoudi-Nejad, A. (2013). PPIevo: Protein-protein interaction prediction from PSSM based evolutionary information. *Genomics*, *102*(4), 237-242.

[102] Zhang, X., Dai, D., & Li, X. Protein complexes discovery based on protein-protein interaction data via a regularized sparse generative network model. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, *9*(3), 857-870.

[103] Zhou, Y., Cheng, H., & Yu, J. X. (2009). Graph clustering based on structural/attribute similarities. *Proceedings of the VLDB Endowment*, *2*(1), 718-729.

[104] Zhou, Y., Cheng, H., & Yu, J. X. (2010). Clustering large attributed graphs: An efficient incremental approach. In *Proceedings of the 2010 IEEE International Conference on Data Mining* (pp. 689-698).