MINING CLUSTERS IN ATTRIBUTED GRAPHS

HE TIANTIAN

Ph.D

The Hong Kong Polytechnic University

2017

The Hong Kong Polytechnic University

Department of Computing

Mining Clusters in Attributed Graphs

HE Tiantian

A thesis submitted in partial fulfilment of the requirements

for the degree of

Doctor of Philosophy

January 2017

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 acknowledgment has been made in the text.

HE Tiantian

ABSTRACT

Many real-world relational data can be modeled as graphs that contain vertices and edges representing, respectively, data entities and their relationship. One of the most important tasks is to discover graph clusters or communities, which are interesting subgraphs in the graph data. To find such clusters in graph data, many computational methods have been proposed. Most of the prevalent approaches discover graph clusters taking into the consideration either different topological properties of the graph, e.g., density, and modularity, or vertex attributes. However, effective computational approaches for discovering clusters in graphs, which consider both topology and attribute as factors are not many. In this thesis, we propose to discover graph clusters using the *Attributed Graph*, which contains a set of vertices, edges, and attributes that are associated with vertices. Combining the edge structure with the attribute, it is possible for a computational method to discover clusters in the attributed graph, taking into the consideration edge structure and attributes. Based on the *Attributed Graph*, we propose four different algorithms. Each of these four algorithms has their unique characteristics and may address the existing challenges in graph clustering. To discover interesting subgraphs in which vertices are inter-related, we propose an algorithm for identifying interesting sub-graphs making use of both edge structure and the degree of attribute association between pairwise vertices (MISAGA). MISAGA formulates the task of discovering k sub-graphs as a constrained optimization problem and solves it by identifying the optimal affiliation of sub-graphs for the vertices through an iterative updating algorithm. In each of the interesting sub-graphs found by MISAGA, vertices are densely connected and their attribute values are significantly associated, although their attribute values might not be the same. As there are no very effective graph clustering algorithms that are based on fuzzy set theory, we propose an algorithm for discovering fuzzy structural patterns in attributed graphs (FSPGA). FSPGA adopts an effective fuzzy clustering framework to allow overlapping clusters to be identified. As the identified clusters in some real applications, e.g., functional modules in biological

graphs, need to be connected components, we further propose two more algorithms, called EGCPI and TBPCI for identifying clusters of interest. Different from other approaches, EGCPI formulates the task of discovering clusters in the attributed graph as an optimization problem and tackles it with evolutionary clustering. It can identify those sub-graphs in which vertices are densely connected as well as their attributes are more similar. TBPCI identifies clusters utilizing local information of vertex connectedness and the attribute association between pairwise vertices in attributed graph. TBPCI may compute the optimal degree of boundedness between each pair of vertices which may capture how strong the vertices can be considered as bounded together. Then the clusters can be identified by grouping those vertices sharing degrees of boundedness which are sufficiently strong. The proposed algorithms have been used in different real-world applications, including community detection in social network graphs and functional modules identification in biological network graphs. The experimental results show these proposed algorithms outperform state-of-the-art approaches.

LIST OF PUBLICATIONS

Journal papers

[1] Tiantian He, Keith C. C. Chan, "Evolutionary graph clustering for protein complex identification," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2016, DOI: 10.1109/TCBB.2016.2642107.

[2] Tiantian He, Keith C. C. Chan, "MISAGA: An Algorithm for Mining Interesting Sub-Graphs in Attributed Graphs," *IEEE Transactions on Cybernetics*, 2017, DOI: 10.1109/TCYB.2017.2693558.

[3] Tiantian He, Keith C. C. Chan, "Discovering Fuzzy Structural Patterns for Graph Analytics," submitted to *IEEE Transactions on Fuzzy Systems* (2nd Round Review).

[4] Tiantian He, Keith C. C. Chan, "Measuring boundedness for protein complex identification in PPI networks," *IEEE/ACM Transactions on Computational Biology and Bioinformatics* (2nd Round Review).

Conference papers

[5] Tiantian He, and Keith C.C. Chan, "Evolutionary community detection in social networks," in *2014 IEEE Congress on Evolutionary Computation (CEC)*, 2014, pp. 1496-1503.

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my supervisor, Professor Keith C. C. Chan. All the time, he offers invaluable advice to my research by his high perspicacity and keeps on encouraging me to overcome the obstacles in the study. Without his considerate guidance and consistent inspiration, this thesis would not have been completed.

I would like to express my thankfulness to Dr. Yang Liu, who always inspires me by sharing his significant experience in academic and life. Also, I would like to thank all my friends and colleges: Prof. Zhuhong You, Dr. Lun Hu, Dr. Xin Luo, Dr. Peiyuan Zhou, Mr. Weimin Luo, Mr. Pengwei Hu, and Mr. Yanxing Hu for the constructive suggestions and illuminating discussions in the last four years.

Special thanks to my parents for all the supports and sacrifices that you have made for me. Your love to me sustained thus far. I would also like to thank my beloved wife Wei Yu. To them, I dedicate this thesis.

Contents

List of Figures

List of Tables

1. INTRODUCTION

Many real relational data can be modeled as graphs containing a set of vertices and edges representing the data entities in the dataset and interconnection between the entities. Compared with the random graphs in which vertices are connected with their neighbors under nearly the same probability, graphs constructed based on real-world data are not random and always possess some special hidden structures [33]. These featured structures are sometimes called *Community Structure* [39]. In graph analytics, typically one of most crucial tasks is to identify community structure containing a number of graph clusters, or communities, which are interesting sub-graphs in which vertices are cohesively inter-related. Such interesting sub-graphs might be named particularly in a specific application. For example, in social network analysis, the discovering of clusters of interest is called social community detection. Similarly, in biological network data, e.g., protein-protein interaction (PPI) networks, the identification of such interesting sub-graphs is named as the identification of functional modules. How to discover such clusters in different types of graph data has drawn much attention in the recent [33] [58]. To discover clusters in graphs, different information can be utilized. Not surprisingly, most proposed approaches detect graph clusters mainly using pre-specific topological properties of the graph, e.g., *edge centrality* [39], *modularity* [76], and edge density. Hence, the graph constructed based on the data contains only vertices and edges, which represent data entities and the inter-connections between them. While, some other algorithms, discover these graph clusters using attribute values associated with the vertices. The discovered clusters contain vertices whose attribute values are more similar. Though effective methods to some extent, these algorithms might overlook some interesting clusters as they do not take into consideration various information stored in the graph data.

In this thesis, we attempt to combine both topology and attribute information that is in the graph and address the challenges existing in the state-of-the-art. Hence, we propose to discover clusters in the *Attributed Graph*, which contains vertices, unidirectional edges, and attributes values, which represent the data entities, the inter-connections between data entities, and features that may describe the entities, respectively.

The rest of this section is organized as the following. In Section 1.1, the challenges existing in the state-of-the-art of graph clustering that may motivate us to propose more effective computational methods are illustrated. In Section 1.2, what sub-graphs are identified as clusters in attributed graphs is introduced. In Section 1.3, the algorithms that may address the challenges are introduced. In Section 1.4, we give the organization of the thesis.

1.1. Motivation

As mentioned, many real applications, such as social community detection, functional module identification, and document segmentation, are trying to find meaningful clusters in the graph constructed by different types of relational data. Thus, the discovering of clusters in graphs has drawn much attention in the recent. And there have been a number of approaches tackling this problem proposed. Most of these proposed algorithms, e.g., CNM [17], BGLL [8], CPM [80], AP [35], Link-Com [1], SC [57], and MMSB [2], may discover clusters in graphs mainly using the pro-specific topological properties of the graph, e.g., edge density, and modularity.

There are some other algorithms which may discover graph clusters only using the attribute values associated with the vertices. For examples, *k*-means [73], MAC [36], and *k*-SNAP [95] can detect meaningful graph clusters based on the degree of similarities computed based on the attribute values associated with the vertices.

Besides, there have been some attempts to detect clusters in the attributed graph. For examples, SA-Cluster [112], Inc-Cluster [113], EDCAR [41], GBAGC [104], CESNA [110], and Circles [74] may discover the clusters in which vertices are densely connected and their attribute values are relatively more similar. Most of these mentioned algorithms may discover disjoint graph clusters only.

Though these mentioned algorithms are effective to some extent, we find the following challenges existing in graph clustering which may motivate us to develop more effective and efficient algorithms.

First, though utilizing different techniques, most algorithms perform the task of discovering clusters taking emphasis more either on some predefined topological properties of the graph or on the similarity of attribute values associated with the vertices. While algorithms that take into consideration both the aforementioned as measures for cluster identification are not many. As a result, there might be some meaningful graph clusters, e.g., social communities in a social network graph, protein complexes in a PPI network graph that cannot be discovered.

Second, though some graph clustering algorithms can discover clusters taking into consideration graph topology and attribute information, the attribute values associated with vertices may not be fully utilized as they consider only those similar attribute values rather than related ones. Hence, some interesting sub-graphs with vertices possessing different but related attribute values may not be identified effectively.

Third, though some algorithms utilize both graph topology and attribute information to identify clusters in a graph, strengths of topology and attributes values that may determine the cluster membership may not be truly revealed. For example, FCAN [44] may detect clusters by segmenting a data matrix in which each element represents the

strength of the relationship between pairwise data points. The entries of the data matrix are obtained by adding the binary value and the degree of similarity representing the connection and attribute similarity between pairwise vertices, respectively. This may also degrade the quality of detected graph clusters.

Forth, most algorithms for detecting graph clusters are partition-based. In other words, they cannot identify overlapping clusters in a given graph. These overlapping sub-graphs might be more desirable in some graph data, e.g., communities in social networks are sometimes overlapping.

Fifth, techniques that can be used for identifying clusters utilizing both topological properties of the graph and attribute information still need developing. For example, there are no effective fuzzy-based algorithms that can identify overlapping clusters in a network graph.

1.2. Problem of clustering in attributed graphs

Different from clusters in graphs containing vertices and edges only, ones in attributed graphs are discovered by taking into the consideration topological and attribute properties. Therefore, how to define sub-graphs to be clusters of interest are the premises of mining clusters in the attributed graph. There have been several measures proposed to define what sub-graphs can be categorized into the clusters. For examples, in model-based algorithms like the ones in [104], [110], and [74], clusters are defined as those sub-graphs in which vertices share the maximum probabilities of being connected and homogeneous attribute values being associated. In density-based methods like the one in [41], clusters are defined as those sub-graphs in which vertices have the relatively high density of inter-connections and homogeneous attribute values. Though different names proposed in different works, sub-graphs which can be clusters of interest can be identified by using measures concerning both topology and attribute

carried by the attributed graph. Hence, in this thesis, we are considering to define the sub-graphs satisfying the following measure of interestingness to be clusters. A sub-graph in an attributed graph is seen as a cluster if its vertices are densely connected and attribute values associated with the vertices are inter-related. It should be noted that inter-related attribute values may not be necessarily the same. Given this definition, vertices in a cluster in the attributed graph are cohesively inter-related not only topologically, but also characteristically. All the methods proposed in this thesis aim at identifying such sub-graphs in the attributed graph as clusters, utilizing different computational techniques.

1.3. An overview of solutions

Given the challenges also the motivations mentioned, we propose to perform the task of mining graph clusters using the *Attributed Graph*. Apparently, the discovering of clusters in an attributed graph is different from that in a traditional graph as attributes are associated with vertices. To discover clusters in an attributed graph, the crucial task is to identify the strength between pairwise vertices resulting from the associated attributes. To fulfill the mentioned task, we propose to identify such strength using different measures. One measure for quantifying the strength of attributes assigned to pairwise vertices is called the *degree of attribute homogeneity*, which is based on the Jaccard Similarity. The *degree of attribute homogeneity* becomes higher when there are more similar attributes associated with pairwise vertices. The other we further propose to use is a probabilistic measure which may identify patterns indicating pairwise attributes are significantly associated. The discovery of significant associations between pairwise attributes may filter out those unrelated attributes associated with pairwise vertices. Utilizing these identified significantly associated attributes, one may reveal the *degree of attribute association* between pairwise vertices. To obtain such degrees, we propose to use either an information theoretic measure [73] or a cosine

similarity measure to compute the total degree of attribute association between pairwise vertices. Combining with the edge structure, it is possible for a computational method to discover clusters in the attributed graph, taking both edge structure and attributes into the consideration.

After transferring a network graph to an attributed graph, we attempt to propose four algorithms to discover clusters, utilizing both graph topology and attribute information. These four algorithms may mine clusters in the attributed graph by making use of different optimizing techniques.

First, we propose MISAGA, which is an algorithm for identifying interesting sub-graphs in attributed graphs making use of both edge structure and the *degree of attribute association* between pairwise vertices. MISAGA formulates the task of discovering clusters as a constrained optimization problem and solves it by identifying the optimal affiliation of sub-graphs for the vertices in the attributed graph through an iterative updating algorithm.

Second, we propose FSPGA, which is an algorithm for detecting clusters in attributed graphs making use of fuzzy clustering. FSPGA adopts an effective fuzzy clustering framework to allow overlapping sub-graphs to be identified, which is very significant in some of the real applications, e.g., community detection in social network analysis.

Third, we propose EGCPI, which is an efficient algorithm for the identification of interesting sub-graphs based on evolutionary clustering. EGCPI formulates the task of discovering clusters in the attributed graph as an optimization problem and tackles it with evolutionary clustering. By using the edge structure and the *degree of attribute homogeneity* measure, it can identify those sub-graphs in which vertices are densely connected, as well as the attributes of vertices, are more similar.

At last, we propose TBPCI, which is an algorithm to identify clusters in attributed graphs taking into the consideration both graph topology and significant attribute associations. TBPCI identifies clusters utilizing local information of vertex connectedness and the *degree of attribute association* between pairwise vertices in attributed graph. TBPCI may compute the optimal *degree of boundedness* between each pair of vertices which may capture how strong the vertices can be considered as bounded together. Then the clusters can be identified by grouping those vertices sharing degrees of boundedness which are sufficiently strong.

The proposed algorithms have been used in different applications, including community detection in social networks and functional modules identification in biological network graphs. The experimental results show these proposed algorithms outperform state-of-the-art approaches.

1.4. Thesis organization

To illustrate how we address the challenges mentioned, we organize the rest of the thesis as the following.

In Section 2, we present the overview of the previous works that are related to detecting clusters in graphs. These related works are categorized based on their features, e.g., topology/attribute based clustering and graph clustering using the techniques of computational intelligence. In addition, we introduce different real problems that can be solved by those mentioned algorithms.

In Section 3, how to represent an attributed graph and how to determine the strength in terms of attribute values between pairwise vertices are introduced. These mentioned issues are used by the proposed algorithms in this thesis.

In Section 4, we present the reason why MISAGA is proposed at first. Then, how MISAGA formulates mining clusters in the attributed graph as a constrained optimization problem and how to solve the problem by using an effective iterative updating algorithm are presented. In addition, the experiments that may test the efficiency and effectiveness of MISAGA and other baselines are presented.

In Section 5, we present the background under which we propose the algorithm FSPGA at first. Then, how FSPGA formulates the discovering of clusters in attributed graphs as a fuzzy optimization problem, and the experiments which may test the efficiency and effectiveness of FSPGA and the compared baselines are introduced.

In Section 6, we present the algorithm EGCPI, which is an evolutionary algorithm for detecting clusters in the attributed graph. The background under which we propose the algorithm, the details of EGCPI, and how to test the effectiveness of EGCPI and other baselines, using the experiments related to the real application, i.e., protein complex identification, are presented.

In Section 7, we present the algorithm TBPCI, which is an algorithm for identifying interesting sub-graphs making use of local information on topology and associated attribute values. The details of the proposed algorithm and how to test the effectiveness of the proposed algorithm and other baselines, using the experiments related to the real application, i.e., functional modules detection in biological graphs, are presented.

At last, in Section 8, we summarize the contributions of the thesis and propose future works.

2. OVERVIEW OF THE RELATED WORK

To discover clusters in graphs, several algorithms have been proposed. Though different computational methodologies might be utilized, these algorithms can be categorized according to the specific properties that are considered, the techniques that are used, and the particular field into which they are applied. In this section, the state-of-the-art related to discovering clusters in graphs are introduced categorically.

2.1. Graph clustering algorithms

To detect meaningful clusters in the graph data, there have been several so-called graph clustering algorithms proposed. These algorithms can be categorized based on the information of graph data they utilize.

2.1.1. Topology-based graph clustering

Unsurprisingly, most algorithms detect graph clusters based on pre-specified topologies or edge structures, such as the vertices being more densely connected within the same cluster than those belong to other clusters. For example, in [39], an algorithm that detects communities based on specific requirements on *edge centrality* is presented. In [76], another measure, called *modularity*, which is defined as a function of the differences in density within and between graph clusters and a null-graph (in which vertices are connected randomly) is proposed. Based on it, several algorithms, such as CNM [17] and BGLL [8], are developed to search for graph clusters based on the optimization of modularity. In [40] and [34], the formalism of a random graph is introduced. This formalism shows clusters smaller than a certain size cannot be detected by these algorithms that detect for clusters based on modularity optimization. Though there are some limitations, these modularity-based algorithms can detect graph clusters in very large network graphs whose size is more than 1,000,000 vertices.

Besides modularity optimization, there are other algorithms that can detect graph

clusters based on other properties of the graph. In [80], for example, an algorithm is proposed to detect graph clusters based on the concept of a clique using a clique percolation method (CPM). In [35], a graph clustering method called affinity propagation (AP) is proposed to detect clusters based on the similarities between cluster centers and other vertices. In [1], a method is proposed to detect graph clusters by introducing the concept of a link graph to facilitate optimization of edge densities. In [57], given vertices in the same cluster may share similar edge structure, spectral clustering is proposed to consider normalized cuts [89] for cluster identification. In [107], a semi-supervised algorithm for detecting clusters in graphs is proposed. Besides of making used of spectral techniques, the proposed algorithm also utilizes the prior knowledge determining the cluster affiliation of some vertices to obtain a better result. In [2] and [53], two approaches based on the Mixed Membership Stochastic Blockmodels (MMSB) are proposed, respectively to detect graph clusters by optimizing the posterior probability that a pair of vertices are actually connected. The higher the probability, therefore, the greater the sub-graph density. There are some algorithms for detecting graph clusters based on other techniques. In [108], a model based algorithm called CoDa is proposed to detect communities in graphs. Modeling the discovering of communities as identifying the community affiliations of each vertex, the best affiliation can be identified by optimizing the posterior probabilities that are used to represent the possibility that vertices belong to a community in a generative model. There are also some graph clustering algorithms that can discover graph clusters based on frequent patterns hidden in the graph, e.g., frequent sub-structure patterns [111]. For example, in [94], an algorithm is proposed to discover graph clusters by grouping the vertices sharing the predefined largest pattern of sub-structure.

2.1.2. Attribute-based graph clustering

Identification of clusters that is based solely on graph topologies does not take into

consideration attribute values associated with the vertices. In the case that such attribute values are useful for the discovering and understanding of the sub-graphs identified, many graph clustering algorithms cannot be used.

To consider attributes in graph clustering, some attempts have been made to make use of the *k*-means algorithm [73] to group vertices with the higher similarity of attributes into the same clusters. In [36], an algorithm (MAC) that is based on probabilistic generative model is proposed for clustering vertices that are labeled with Boolean attribute values. In [95], a graph summarization algorithm called *k*-SNAP is proposed to detect graph clusters by grouping vertices into the same cluster according to a similarity measure of the attribute values.

2.1.3. Graph clustering using graph topology and attribute

Those graph clustering algorithms that take into consideration either topological properties of the graph or attributes associated to vertices are not very well suited for the task of discovering meaningful clusters in the graph because they take more emphasis either on graph topology or attributes associated with the vertices, but overlook the other.

To consider both attributes and structures in graph clustering, several algorithms can be used. In [112], SA-Cluster is proposed to detect graph clusters using a neighborhood random walk model. Based on it, cluster membership of each vertex is determined at the time that the transition matrix reaches steady-state. In [113], inc-Cluster is proposed using the same random walk model as the SA-Cluster except that its efficiency is improved using an incremental method to compute the transition matrix. In [41], EDCAR is proposed to mine sub-graphs by grouping together vertices that are densely connected and share similar attribute values. In addition to graph structures, these algorithms are able to take into consideration vertex attributes. However, they are more

graph partitioning algorithms that are not developed to discover overlapping sub-graphs. Thus, interesting graph clusters may sometimes be overlooked.

In addition to the above algorithms, some algorithms detect graph clusters by utilizing generative models. In [104], a general Bayesian model for graph clustering (GBAGC) is proposed to make use of a Bayesian generative model to estimate structural and attribute similarity of pairwise vertices in each cluster. A number of disjoint graph clusters are obtained after all parameters are estimated. In [110], an algorithm, called CESNA, is proposed to make use of a statistical model to determine the posterior probability that pairwise vertices are connected given particular edge structures and attributes in a graph cluster. Cluster membership is determined when the posterior probability is maximized. In [74], an algorithm called Circles is proposed to detect clusters in social network graphs. By taking user profiles as attributes, Circles determines cluster membership by estimating the similarity between user attributes and those which are commonly observed in members of each cluster. The cluster membership of a vertex is determined to be those that are predicted to have higher similarities with other vertices in the same clusters. In [44], FCAN is proposed to discover graph clusters in the complex network by using both link structure and relevant content associated with vertices. FCAN may detect clusters by segmenting a data matrix in which each element represents the strength of the relationship between pairwise data points. The entries of the data matrix are obtained by adding the binary value and the degree of similarity representing the connection and attribute similarity between pairwise vertices, respectively.

Inspired by topic modeling [10], several topic-model-based approaches, such as Link-PLSA-LDA [77], Relational Topic Model [18], iTopicModel [90], PL-DC [109] and Block-LDA [11] can also be used to identify graph clusters mainly in document

networks with words as attributes of vertices and citations as edges of an attributed graph. With these topic-model-based approaches, cluster membership is determined by maximizing the probability that vertices in the same cluster labeled with the same topics. However, due to rather high demand for computational resources, these topic-model-based approaches are not developed to handle large attributed graphs [110].

2.2. Detecting graph clusters using computational intelligence

Besides those mentioned graph clustering algorithms, there have been several algorithms which may detect graph clusters based on different techniques of computational intelligence. Generally speaking, evolutionary computation and fuzzy set theory have been used to develop the approaches to detecting clusters in graph data.

2.2.1. Evolutionary graph clustering

As the problem of graph clustering can be formulated as an optimization problem with relatively clear objectives, evolutionary algorithms (EAs) have been used to tackle the problem. Recently, there have been some effort to detect community structures using genetic algorithms (GAs) and the approach has been shown to be very effective.

The first successful GA application for community detection is described in [96]. The approach makes use of the modularity measure as the fitness function so that it can be optimized. To eliminate communities with relatively lower modularity scores and uniqueness at the end of each reproduction cycle, an information theoretical measure is used to eliminate communities with relatively lower modularity scores and uniqueness. As a result, the quality of the final communities detected can be enhanced.

Another GA developed to detect community structures was proposed in [59]. It takes into consideration Silhouette Width [86], normalized cut [89] and the modularity measure in a fitness function used to guide the evolutionary process. To facilitate the

exchange of community structures, this GA also makes use of a special crossover operator. Even though the operator slows down the evolutionary process, it allows for a greater diversity of community structures.

Other than detecting graph clusters using single-objective evolutionary algorithms, multi-objective evolutionary algorithms can also be used for detecting graph clusters. For example, a multi-objective GA for detecting communities is proposed in [81]. Other than the modularity measure, the fitness of a community structure is evaluated based also on the use of another benchmark called community fitness [60]. Partitioning of a complex network has to allow both the two objectives to be optimized. Another example of multi-objective EA for graph clustering is proposed in [61], which is called MEAs-SN. MEAs-SN is a multi-objective evolutionary algorithm which can detect social clusters with more intra-positive edges as well as more negative edges between clusters. In [19], another evolutionary algorithm (DCRO) for detecting graph clusters is proposed. Different from MEAs-SN, DCRO detects graph clusters based on an efficient multi-objective evolutionary algorithm called Chemical reaction optimization (CRO) [12].

Besides the above evolutionary algorithms, there have been some attempts to detect graph clusters considering both network topology and attribute information. For example, an evolutionary community detection algorithm, called ECDA [45], is proposed to detect for communities in social networks by considering network connections and attribute labeled to each pair of vertices.

2.2.2. Fuzzy graph clustering

Besides above approaches, algorithms based on fuzzy techniques can also be applied to find graph clusters taking into consideration attributes associated to the vertices. For example, the classical fuzzy c-means algorithm [9] can discover graph clusters by

grouping those vertices with similar attributes together. In addition, other fuzzy clustering algorithms which are based on the fuzzy c-means model, such as relational fuzzy c-means algorithm [43] and improved fuzzy c-means algorithm [54], can also be used to detect clusters given the similarity of attribute values between pairwise vertices.

Though adopting different techniques to detect graph clusters, most existing evolutionary algorithms for detecting graph clusters make use of various topological properties of networks when performing their tasks. While, those fuzzy algorithms for clustering in graphs mainly perform the task by taking into the consideration the similarity of attribute values between pairwise vertices. Thus, they also can be seen as graph clustering algorithms based on either graph topology or attribute values.

2.3. Graph clustering in real applications

As mentioned above, there are several real problems that can be solved as clustering in graph data. Given the state-of-the-art, two dominant applications are always investigated, including social community detection and functional module detection in biological graphs. Based on the characteristics of the applications, algorithms that are applied into real problems might be slightly different.

2.3.1. Social community detection

Social community detection is one of the most important applications of social network analysis. Many algorithms are proposed to solve the problem of community detection in social graphs. For examples, CoDa [108], Circles [74], ECDA [45] are three algorithms for detecting communities in social graphs. Besides, algorithms like CNM [17], BGLL [8], SA-Cluster [112], inc-Cluster [113], EDCAR [41], and CESNA [110] can also be used for social community detection. It is said that, most graph clustering algorithms are capable of performing the task of discovering meaningful communities in social graphs.

2.3.2. Graph clustering in PPI networks

Recently, protein complex (functional module) identification in protein-protein interaction networks has also drawn much attention due to its important role in the understanding of cellular organizations and functions, such as replication, transcription and the control of gene expression, etc. [46]. A protein complex is a biomolecule that contains a number of proteins connecting with each other to perform cellular functions [91]. Based on the definition of the protein complex, it is also a sub-graph of interest existing in the PPI network graph. Due to the uniqueness of protein complex identification, e.g., proteins in a protein complex are always connecting with each other, algorithms for protein complex identification might not always the same to those general graph clustering algorithms. Thus, the state-of-the-art for protein complex identification should be investigated separately.

To identify protein complexes on a large scale, time-consuming laboratory experiments, such as affinity purification (AP) followed by mass spectrometry (MS), have to be performed [38], [46]. Though effective, AP/MS cannot be considered as an efficient method as it requires a number of different steps to be carried out with different baits every time [28].

To minimize the laborious trials-and-errors procedures, some attempts to identify protein complexes computationally have recently been made [51] [62]. Most of these computational methods are developed based on different graph clustering algorithms. Due to some evidence of proteins in protein complexes tending to interact more with each other, many algorithms for protein complex identification aim at identifying densely connected sub-graphs by considering some graph properties, e.g., modularity and density, as protein complexes [87], [97], [114]. For example, one of the most popular graph clustering algorithms that are used for protein complex identification is MCODE [13]. MCODE can identify sub-graphs that contain densely connected vertices

in a PPI network graph as protein complexes, by taking into consideration local neighborhood density.

Other than MCODE, another graph clustering algorithm called the MCL algorithm [29] has also been used for the identification of protein complexes. The MCL algorithm also discovers densely connected sub-graphs except that it does so by making use of a random-walk approach through simulating flow expansion and contraction [30] using what are called expansion and inflation operators. A number of dense clusters can be extracted from the incidence matrix of a PPI network graph when MCL achieves convergence.

Another dense sub-graph identification algorithm, called RNSC [55] is proposed to identify protein complexes in PPI network graphs through graph partitioning. RNSC attempts to find an optimal set of partitions of a PPI network graph by employing different cost functions that are defined in terms of edge density, cluster size, and functional homogeneity. The graph partitions that are identified can correspond well with protein complexes. An algorithm that is similar to the RNSC is proposed in [31]. The algorithm also attempts to find protein complexes by partitioning a PPI network graph but it uses a minimum vertex-cut to identify cluster boundaries so that the vertices in each graph partition tend to connect more with other vertices that are in the same partition.

In [52], an algorithm called SPICi is proposed to identify protein complex by considering the local density of a PPI network. Comparing with other density-based graph clustering algorithms, SPICi can be shown to be a very fast algorithm for protein complex identification. In [115], an algorithm called DCU is proposed to detect protein complexes utilizing an uncertain graph model. DCU utilizes two measures when identifying protein complexes. One of them is called the relative degree measure. It is

used to determine whether a protein belongs to a sub-graph. The other is called the expected density measure. It is used to determine whether a dense sub-graph satisfies with the minimum density to be identified as a protein complex.

Other than edge density, some graph clustering algorithms perform their tasks by considering other graph properties and some of these algorithms have also been used for protein complexes identification in PPI networks. For example, a graph clustering algorithm called DPClus [3], can discover and refine graph clusters by keeping track of cluster periphery. This DPClus algorithm is later improved to enhance computational efficiency in another algorithm called IPCA [63]. IPCA finds graph clusters based on a vertex distance and a density measure.

Another example of an algorithm that finds graph clusters based on graph properties other than edge-density is CFinder [4]. CFinder identifies graph clusters based on clique percolation and the graph clusters identified correspond well to known protein complexes. Similar to CFinder, the CMC algorithm also finds graph clusters based on the discovering of cliques [64]. By iteratively assigning weights which indicate the reliability of interactions between proteins, CMC attempts to find cliques that have the highest value of such weights. Other than CMC, another clique-based algorithm called IPC-MCE is proposed in [65]. The algorithm takes each maximal clique identified as the core of a protein complex. It then extends the core by including those “peripheral” proteins that are determined to have a higher probability to connect to the core.

In [100], an algorithm called COACH is proposed to find protein complexes making use of a different graph property, i.e. core-attachment. In [82], another core-attachment-based algorithm called WPNCA is proposed. Different from COACH, WPNCA utilizes a Pagerank-nibble algorithm to assign a weight to each interaction in a PPI network to obtain a better performance.

Most graph clustering algorithms do not discover overlapping clusters and cannot be used to identify overlapping protein complexes. There are some exceptions, however. For example, an algorithm proposed in [116] can detect overlapping protein complexes based on a generative network model. Another algorithm called ClusterONE [78] can also do so using a measure called graph cohesiveness.

As there is some evidence of proteins belonging to the same protein complex performing similar or related functions [101], there are also some attempts to identify protein complexes that can take into consideration information about protein attributes rather than topology only. For example, in [66], PCIFI identifies protein complexes based on finding connected proteins that perform interdependent molecular functions. In [67], another algorithm performs the task of protein complex identification by simultaneously using PPI network data and gene expression data. In [47], an algorithm called PCIA is proposed to identify protein complexes in PPI networks based on network topology and attribute information. It makes use first of a measure of attribute similarity followed by the use of the MCL algorithm to identify densely connected clusters during the process. In [117], an algorithm called GMFTP, is proposed to identify protein complexes based on measures of similarity between attribute values of proteins. Given the experimental results presented, all the mentioned approaches have shown their effectiveness in protein complex identification.

According to the experimental results shown in the related publications, all these mentioned approaches have shown their effectiveness in one or several real applications of detecting clusters in graphs. Given the challenges summarized in Section 1, we will propose four different algorithms, i.e., MISAGA, FSPGA, EGCPI and TBPCI to address them accordingly.

3. TECHNICAL PRELIMINARIES

To mine clusters, it is essential to construct an attributed graph given a set of data containing data entities, inter-connections, and features describing the data entities, and determine how strong inter-related the pairwise vertices are regarding of the attribute values. In this section, the notations of the attributed graph and the determination of the strength in terms of the attribute values between pairwise vertices are introduced.

3.1. Notation of the attributed graph

Given a set of relational data which contains vertices, edges, and a set of attributes representing the data entities, relationship between entities and characteristics that may describe the entities, it can be represented as an *Attributed Graph*, containing n_V vertices and n_E edges, in which each vertex is associated with a set of attribute values, the graph can be represented as G = (V, E, Λ), where the set of vertices, V, can be denoted as, V= $\{v_i \mid 1 \leq i \leq n_V\}$, the set of edges, E, can be denoted as E = $\{e_{ij} \mid 1 \leq i, j \leq n_V, i \neq j\}$, and the set of attributes that is associated with each vertex can be denoted as Λ where Λ = $\{att_i \mid 1 \leq i \leq n_\Lambda\}$. The algorithms proposed in this thesis will detect clusters of interest by representing relational data as G.

3.2. Attribute strength between pairwise vertices

To discover clusters of interest in G, it is important to determine how strong a pair of vertices are inter-related, according to the attribute values associated. To fulfill the task, we propose two different measures to determine such strength between pairwise vertices in the attributed graph.

One measure is called the *degree of attribute homogeneity* (θ), it is defined as

$$\theta_{ij} = \frac{|\Lambda_i \cap \Lambda_j|}{|\Lambda_i \cup \Lambda_j|} \tag{1}$$

where Λ_i represents the set of attribute values associated with vertex v_i in G. θ is a Jaccard Similarity measure and ranges from 0 to 1. Apparently, θ becomes higher when there are more homogeneous attributes values associated with both pairwise vertices in G. Hence, θ may determine how similar the attribute values associated with pairwise vertices.

The other measure is called the *degree of attribute association* (φ). φ can be obtained following a two-step approach. First, patterns of significant associated attribute values in G can be obtained by making use a statistical measure. To illustrate how this can be done, let us consider two attribute values, att_i and att_j. Let us use o(att_i, att_j) to denote the frequency that att_i and att_j appears respectively in the attribute value sets of two vertices which are connected. To denote the expected frequency that att_i and att_j are connected in G, we use the notation e(att_i, att_j). We consider att_i and att_j as having a significant interesting association with each other if o(att_i, att_j) is sufficiently different from e(att_i, att_j). To determine if the difference is statistically significant, we make use of the following test statistics

$$diff\,(att_i, att_j) = \frac{o(att_i, att_j) - e(att_i, att_j)}{\sqrt{e(att_i, att_j)(1 - \frac{o(att_i+)}{n_E})((1 - \frac{o(att_j+)}{n_E}))}} \tag{2}$$

where o(att_i+) represents the frequency that attribute value att_i is associated with any connecting vertices. In [20], [21] and [48], this measure is shown to approximately follow the *Standard Normal* distribution. One may, therefore, decide that att_i and att_j are significantly associated with each other at a 95% confidence level if *diff*(att_i, att_j) is greater than 1.96. Otherwise, they can be considered not significantly associated with each other. With this measure, those attribute values that are not sufficiently relevant are filtered out. Although two attribute values are different, they might be significantly

relevant if *diff* between them is larger than some threshold.

Having obtained the significantly associated patterns in attribute values, one may determine the *degree of attribute association* between two vertices, say v_i and v_j, by using appropriate measures. In this thesis, we mainly using two different measures tow compute φ between pairwise vertices in G. One method for computing φ is based on an information theoretic measure [73]. Let att_{ik} and att_{jm} denote a pair of associated attribute values belonging to the attribute sets of v_i and v_j respectively. Let $\Pr(att_{ik}, att_{jm})$ denotes the probability that att_{ik} and att_{jm} are connected in G. Let $\Pr(att_{ik})$ and $\Pr(att_{jm})$ denote the probability that att_{ik} and att_{jm} are associated with the connected vertices respectively, then the total *degree of attribute association* between v_i and v_j is defined as

$$r(v_i, v_j) = \sum_k \sum_m \Pr(att_{ik}, att_{jm}) \cdot \log \frac{\Pr(att_{ik}, att_{jm})}{\Pr(att_{ik}) \cdot \Pr(att_{jm})} \tag{3}$$

It should be noted that the magnitude of $r(v_i, v_j)$ increases with the number of associations between attribute values and the relative magnitude of the probabilities which reflect the degree of association between them. As those attribute values that are not significantly associated with each other has been filtered out, we only consider relevant and interesting attribute values. For the purpose of normalization, MISAGA computes an entropy measure as follows:

$$H(v_i, v_j) = -\sum_k \sum_m \Pr(att_{ik}, att_{jm}) \cdot \log \Pr(att_{ik}, att_{jm}) \tag{4}$$

Given (3) and (4), φ between the vertices v_i and v_j can be obtained as

$$\varphi_{ij} = \frac{r(v_i, v_j)}{H(v_i, v_j)} \tag{5}$$

Using the above computational method, φ_{ij} can be interpreted as the information redundancy of the attribute values that are associated with v_i and v_j. After normalization, the magnitude of φ_{ij} ranges from 0 to 1. A greater value of that means that the attribute values of the pair of vertices, v_i and v_j are more strongly associated with each other, and so, their being members in the same group may reflect more interesting patterns.

The other method for computing φ is based on cosine similarity, given all *diff*s between two vertices, v_i and v_j we can determine whether pairwise attributes values are significantly associated as (6) shows:

$$diff\,(a_{ik}, a_{jm}) = \begin{cases} 1, if\ \ diff\,(att_{ik}, att_{jm}) \geq 1.96 \\ 0, otherwise \end{cases} \tag{6}$$

Once whether or not a combination of attributes values is significantly associated has been decided, we can compute φ between each pair of vertices in G

$$\varphi_{ij} = \frac{\sum_k \sum_m diff\,(att_{ik}, att_{jm})}{|\Lambda_i| \times |\Lambda_j|} \tag{7}$$

A higher value of φ means a higher proportional pairwise combination of significantly associated attribute values existing between two vertices. And it is concluded that the two vertices have a stronger association when considering their attribute information. By utilizing the θ measure and φ measure, the proposed algorithms in this thesis may determine how strong the pairwise vertices are inter-related in G when considering the attribute values associated. Combining with the edge structure, the proposed approaches may discover meaningful clusters taking into the consideration both graph

topology and attribute information in the attributed graph data.

4. MISAGA-AN ALGORITHM FOR MINING CLUSTERS IN ATTRIBUTED GRAPHS BY IDENTIFYING OPTIMAL CLUSTER MEMBERSHIP

4.1. Background

As mentioned before, clusters in graphs can be seen as interesting sub-graphs. In this section, an algorithm for identifying interesting sub-graphs in attributed graphs, MISAGA, is proposed. Taking into the consideration both edge structure and significantly associated attribute values of pairwise vertices in the attributed graph, MISAGA identifies interesting sub-graphs by formulating it as a constrained optimization problem and solves it by identifying the optimal affiliation of sub-graphs for the vertices in the attributed graph. MISAGA has been tested with several large-sized real graphs and is found to be potentially very useful for various applications.

The identification of interesting sub-graphs utilizing both graph topology and attribute information has been a widely accepted scenario. There have been several algorithms proposed to solve the problem taking into consideration edge structure and attribute information. For examples, SA-Cluster [112] and Inc-Cluster [113] are two random-walk-based algorithms that can detect disjoint clusters in the attributed graph utilizing both connections and similar attribute values between pairwise vertices. EDCAR [41] is proposed to mine sub-graphs by grouping together vertices that are densely connected and share similar attribute values. GBAGC [104], CESNA [110], Circles [74] are three model-based algorithms for detecting clusters in the attributed graph. Link-PLSA-LDA [77], Relational Topic Model [18], iTopicModel [90], PL-DC [109] and Block-LDA [11] are approaches to partition relational data into clusters utilizing the topic models [10]. All these mentioned methods have shown to be effective.

Having reviewed most algorithms for sub-graphs identification in attributed graphs, we have the following findings that may motivate us to propose a more effective approach. First, algorithms that consider both graph topology and attribute information are not

many. As mentioned in Section 1, most graph clustering algorithms utilize pre-specified topologies or attribute information. Second, almost all the algorithms utilize attribute similarity when detecting clusters. Thus, relevant attribute values which are different are overlooked within the clustering process. This may miss some interesting sub-graphs hidden in the attributed graph. Third, the efficiency of the state-of-the-art needs improving to deal with large graphs. For examples, those topic-model-based approaches are not able to segment large attributed graphs [110].

To address the mentioned challenges, we propose MISAGA (Mining Interesting Sub-graphs in Attributed Graph Algorithm). MISAGA performs its tasks by using a statistical measure that identifies attribute values that have interesting associations with each other. If the attribute values of a pair of vertices have interesting associations, then a measure called the *degree of attribute association* is computed. With such measures, the problem of discovering interesting sub-graphs is formulated and solved as a constrained optimization problem, MISAGA can find an optimal sub-graph arrangement by taking into consideration both edge structure as well as the attribute values associated with each vertex by so doing.

For performance evaluation, MISAGA is tested with both synthetic and real data sets including social and PPI network data. The experimental results are verified against known ground-truth data. The findings show that the sub-graphs discovered by MISAGA can be very meaningful.

4.2. MISAGA in details

4.2.1. Mathematical preliminaries

Given an attributed graph containing n_V vertices and n_E edges, in which each vertex is associated with a set of attribute values, MISAGA models the attributed graph as $G = (V, E, \Lambda)$, which has been shown in Section 3.

Given G, MISAGA constructs an adjacency matrix **Y** of dimensions, n_V by n_V, to represent the connections between vertices in G so that an entry, y_{ij}, in **Y** has the value, 1, if v_i and v_j are connected and, 0, if they are not.

After obtaining **Y** which is used for representing the edge information of G, MISAGA uses another n_V-by-n_V matrix, **A**, to represent the mutual strength in terms of attribute values associated with the pairwise vertices. Here, each entry of **A**, say a_{ij}, is obtained by using the φ measure shown in (2), (3), (4), and (5). Hence, **A** can be named as *attribute-association matrix*. Using **Y** and **A**, MISAGA searches for an optimal solution to a constrained optimization problem that considers both edge structure and attribute association for the mining of interesting sub-graphs as clusters in attributed graphs.

Having obtained the adjacency matrix **Y** and the *attribute-association matrix* **A**, MISAGA will detect sub-graphs existing in G by utilizing the information in the above two matrices. In this section, how MISAGA formulates the mining of sub-graphs as an optimization problem and how MISAGA solves the problem is illustrated.

4.2.2. The objective function for mining interesting sub-graphs

To formulate the sub-graph identification problem as an optimization problem, let us define k to be the number of interesting sub-graphs in G that is to be identified. Given k, we first introduce two n_V-by-k auxiliary matrices, **D** and **B**, which represent the strength that each vertex belongs each of the k sub-graph, taking into consideration structure and attribute associations, respectively. With **D** and **B,** we can introduce a sub-graph membership matrix **C**, which has the dimension of n_V by k. Each element of **C**, say c_{ij}, indicates the strength of membership for vertex i to belong to sub-graph j so that the larger the value of c_{ij}, the greater the affinity between vertex i and sub-graph j.

Given the adjacency matrix **Y**, the attribute-association matrix, **A**, the auxiliary

matrices **D** and **B**, and the sub-graph membership matrix **C**, MISAGA attempts to maximize the following objective function:

$$\begin{aligned}&\text{maximize}\\&O=\alpha tr(\mathbf{C}^{\mathbf{T}}\mathbf{YD})+(1-\alpha)tr(\mathbf{C}^{\mathbf{T}}\mathbf{AB})\\&-\frac{1}{2}\left[\left|\mathbf{C}\right|_F^2+\left|\mathbf{D}\right|_F^2+\left|\mathbf{B}\right|_F^2+\left|\mathbf{BC}^{\mathbf{T}}\right|_F^2+\left|\mathbf{DC}^{\mathbf{T}}\right|_F^2+\lambda\left|\mathbf{Ce_1}-\mathbf{e_2}\right|_F^2\right]\\&\textit{subject to}\ \mathbf{C}\geq 0,\ \mathbf{D}\geq 0,\ \mathbf{B}\geq 0\end{aligned} \tag{8}$$

where (i) $|\mathbf{C}|^2_F$, $|\mathbf{D}|^2_F$, and $|\mathbf{B}|^2_F$ are the F-norm of matrices **C**, **D**, and **B**, which are used respectively to smooth the variables in these matrices, (ii) $|\mathbf{DC}^{\mathrm{T}}|^2_F$, and $|\mathbf{BC}^{\mathrm{T}}|^2_F$ are the F-norm of the products of **D** and the transpose of **C**, and **B** and the transpose of **C**, respectively, (iii) α is a parameter that is used to adjust the relative weighting between edge density and attribute association to be considered when MISAGA searches for the optimal **C**, (iv) $\mathbf{e_1}$ and $\mathbf{e_2}$ are k-by-1 and n_V-by-1 vectors in which all elements are set to 1, (v) $|\mathbf{Ce_1}-\mathbf{e_2}|^2_F$ is an l_2-norm which is used to regulate the aggregation of the variables in each row of **C** so that they can be approximated by 1, which can be used conveniently when comparing the strength of sub-graph affiliation between different vertices. (vi) λ is a non-negative factor which is used to control the effects of the regulation term mentioned in (v). By utilizing the proposed objective function, MISAGA possesses the following advantages when used to identify interesting sub-graphs in attributed graphs.

First, by introducing the two auxiliary matrices **D** and **B**, MISAGA can identify interesting sub-graphs by taking into consideration both graph topology and attribute association between the pairwise vertices in a graph. To explain how MISAGA determines the sub-graph membership of the vertices, let us consider the first two terms of the objective function: $tr(\mathbf{C}^{\mathbf{T}}\mathbf{YD})$ and $tr(\mathbf{C}^{\mathbf{T}}\mathbf{AB})$.

These terms are used to aggregate the strength of the topology and attribute association

of all sub-graphs respectively. Each element in $tr(\mathbf{C}^{\mathbf{T}}\mathbf{YD})$ is used to aggregate the total number of edges within a sub-graph, given the values of the entries in **C** and **D.** $tr(\mathbf{C}^{\mathbf{T}}\mathbf{YD})$ can be optimized only when all the vertices can be assigned to appropriate sub-graphs in which they are connected by more intra-edges. In other words, the corresponding entries in **C** and **D**, say c_{ij} and d_{ij}, which are used to represent sub-graph membership, and the structural strength for vertex i to belong to sub-graph j, should be relatively high when vertex i are connected by relatively more vertices in sub-graph j.

In the case of $tr(\mathbf{C}^{\mathbf{T}}\mathbf{AB})$, it aggregates the total degree of associations between pairwise vertices within each sub-graph. $tr(\mathbf{C}^{\mathbf{T}}\mathbf{AB})$ can be optimized only when all the vertices are assigned to the sub-graphs in which the degrees of attribute association between the pairwise vertices are high.

Given the objective function as shown in (8), MISAGA attempts to find a sub-graph membership matrix **C**, that can be used to assign each vertex into a sub-graph in such a way that it is more connected to and shares a higher degree of attribute association with the other vertices in the same sub-graph. If such an optimal sub-graph assignment can be found, the optimization process adopted by MISAGA can find corresponding optimum values in the entries for this vertex in **D** and **B** and as a result, the corresponding element in **C**, which is used to represent the sub-graph membership between that vertex and that sub-graph is also at its optimum.

However, one may notice that $tr(\mathbf{C}^{\mathbf{T}}\mathbf{YD})$ and $tr(\mathbf{C}^{\mathbf{T}}\mathbf{AB})$ also increase when the variables in **D**, **B**, and **C** simply increase. Thus, we use $|\mathbf{DC}|^2_F$, and $|\mathbf{BC}|^2_F$ to penalize improper variations of the variables in **D**, **B**, and **C** so that, only when the elements of **D**, **B**, and **C** are assigned with appropriate values, the objective function O can be maximized. In such case, **C** contains the membership of the most interesting sub-graphs in each of which vertices are densely connected and their attribute values are significantly

associated.

The other advantage that the optimization process adopted by MISAGA is that, by using the penalty term $\lambda|\mathbf{Ce_1}-\mathbf{e_2}|^2_F$, the aggregation of each row in **C** can be controlled to be around 1 and this can make comparison of the strength of sub-graph affiliation between different vertices and the extracting of overlapping sub-graphs to be more convenient. With these advantages, therefore, when an optimal value for the proposed objective function can be determined, the sub-graphs found by MISAGA can take both edge structure and attribute association into consideration at the same time.

4.2.3. The iterative updating algorithm

The proposed objective function is a constrained quadratic function. It is non-convex to **C**, **D**, and **B** simultaneously, but it is convex to **C**, **D**, or **B** when keeping the other two matrices unchanged. Based on this property, a series of updating rules for optimizing (8) can be obtained.

4.2.3.1. *Updating rule for **C***

Let β_{ij} be the Lagrange multipliers for the constraints $c_{ij} \geq 0$. The Lagrange function L for **C** is

$$L(\mathbf{C},\boldsymbol{\beta}) = O - tr(\boldsymbol{\beta}^{\mathbf{T}}\mathbf{C}) \tag{9}$$

where $\boldsymbol{\beta} = [\beta_{ij}]$ is the matrix of Lagrange multipliers for the non-negativity of **C**. Based on the KKT condition for constrained optimization, we have the following

$$\begin{aligned} &\frac{\partial L}{\partial \mathbf{C}} = \alpha\mathbf{YD} + (1-\alpha)\mathbf{AB} + \lambda\mathbf{e_2}\mathbf{e_1^T} - \mathbf{CD^TD} - \mathbf{CB^TB} - \lambda\mathbf{Ce_1}\mathbf{e_1^T} - \mathbf{C} - \boldsymbol{\beta} = 0 \\ &\boldsymbol{\beta} \circ \mathbf{C} = \mathbf{0} \\ &\boldsymbol{\beta} \geq 0 \end{aligned} \tag{10}$$

where "$\circ$" means the Hadamard product of two matrices with the same dimension. Based on (10), we have the element-wise equation system for each element in **C**

$$\begin{aligned}
&[\alpha\mathbf{YD}+(1-\alpha)\mathbf{AB}+\lambda\mathbf{e_2}\mathbf{e_1^T}]_{ij}-(\mathbf{CD^TD}+\mathbf{CB^TB}+\lambda\mathbf{Ce_1}\mathbf{e_1^T}+\mathbf{C})_{ij}-\beta_{ij}=0\\
&\beta_{ij}\circ c_{ij}=0\\
&\beta_{ij}\geq 0
\end{aligned} \tag{11}$$

Given the first equation in (11), we have

$$[\alpha\mathbf{YD}+(1-\alpha)\mathbf{AB}+\lambda\mathbf{e_2}\mathbf{e_1^T}]_{ij}-(\mathbf{CD^TD}+\mathbf{CB^TB}+\lambda\mathbf{Ce_1}\mathbf{e_1^T}+\mathbf{C})_{ij}=\beta_{ij} \tag{12}$$

Using (12) to replace β_{ij} in the equation of Hadamard product, we have the iterative updating rule for **C**

$$c_{ij}\leftarrow c_{ij}\frac{(\alpha\mathbf{YD}+(1-\alpha)\mathbf{AB}+\lambda\mathbf{e_2}\mathbf{e_1^T})_{ij}}{(\mathbf{CD^TD}+\mathbf{CB^TB}+\lambda\mathbf{Ce_1}\mathbf{e_1^T}+\mathbf{C})_{ij}} \tag{13}$$

4.2.3.2. *Updating rule for* ***D***

Let γ_{ij} be the Lagrange multipliers for the constraints $d_{ij}\geq 0$, hence the Lagrange function L for **D** is

$$L(\mathbf{D},\boldsymbol{\gamma})=O-tr(\boldsymbol{\gamma}^{\mathbf{T}}\mathbf{D}) \tag{14}$$

where $\boldsymbol{\gamma}=[\gamma_{ij}]$ is the matrix of Lagrange multipliers for the non-negativity of **D**. Based on the KKT condition, we have

$$\begin{aligned}
&\frac{\partial L}{\partial\mathbf{D}}=\alpha\mathbf{YC}-\mathbf{DC^TC}-\mathbf{D}-\boldsymbol{\gamma}=0\\
&\boldsymbol{\gamma}\circ\mathbf{D}=\mathbf{0}\\
&\boldsymbol{\gamma}\geq 0
\end{aligned} \tag{15}$$

Given the equation system (15), the element-wise updating rule for **D** can be derived

$$d_{ij} \leftarrow d_{ij} \frac{(\alpha \mathbf{YC})_{ij}}{(\mathbf{DC^T C} + \mathbf{D})_{ij}} \tag{16}$$

4.2.3.3. *Updating rule for **B***

Let η_{ij} be the Lagrange multipliers for the constraints $b_{ij} \geq 0$, hence the Lagrange function L for **B** is

$$L(\mathbf{B}, \boldsymbol{\eta}) = O - tr(\boldsymbol{\eta}^{\mathbf{T}} \mathbf{B}) \tag{17}$$

where $\boldsymbol{\eta} = [\eta_{ij}]$ is the matrix of Lagrange multipliers for the non-negativity of **B**. Based on the KKT condition, we have

$$\begin{aligned} &\frac{\partial L}{\partial \mathbf{B}} = (1-\alpha)\mathbf{AC} - \mathbf{BC^T C} - \mathbf{B} - \boldsymbol{\eta} = 0 \\ &\boldsymbol{\eta} \circ \mathbf{B} = \mathbf{0} \\ &\boldsymbol{\eta} \geq 0 \end{aligned} \tag{18}$$

Given the equation system (18), the element-wise updating rule for **B** can be derived

$$b_{ij} \leftarrow b_{ij} \frac{[(1-\alpha)\mathbf{AC}]_{ij}}{(\mathbf{BC^T C} + \mathbf{B})_{ij}} \tag{19}$$

While keeping other matrices unchanged, these updating rules will guide **C**, **D**, and **B** to identify the local optima in each iteration, respectively.

4.2.4. Convergence analysis of the proposed updating rules

To prove the convergence of the algorithm, we may make use of one property of an auxiliary function that is also used in the proof of the Expectation-Maximization

algorithm [32]. The property of the auxiliary function is described as the following. If there exists an auxiliary function satisfying the conditions that $Q(x, x') \leq F(x)$ and $Q(x, x) = F(x)$, then F is non-decreasing under the updating rule that

$$x^{t+1} = \arg\max_{x} Q(x, x') \tag{20}$$

The equality $F(x^{t+1}) = F(x^t)$ holds only when x is a local maximum of $Q(x, x')$. By iteratively updating x according to (20), F will converge to the local maximum $x_{max} = \text{argmax}_x F(x)$. By defining an appropriate auxiliary function for O, we may show the convergence of (8).

First, we may prove the convergence of the updating rule (13). Let c_{ij} be any element in **C**, O_{cij} be the partial of (8) that is related to c_{ij}, $O_{cij}(c'_{ij})$ be the partial objective value of (8) that is related to c_{ij} when c_{ij} is equal to some value, say c'_{ij}. Since the updating rule for **C** is element wise, it is sufficient to show O_{cij} is non-decreasing according to the updating rule (13). To prove this, we define the following auxiliary function for O_{cij}:

$$Q(c_{ij}, c_{ij}^t) = O_{c_{ij}} + O'_{c_{ij}}(c_{ij} - c_{ij}^t) - \frac{(\mathbf{CD^TD} + \mathbf{CB^TB} + \lambda \mathbf{Ce_1e_1^T} + \mathbf{C})_{ij}}{2c_{ij}^t}(c_{ij} - c_{ij}^t)^2 \tag{21}$$

where O'_{cij} is the first order partial derivative relevant to c_{ij}. Although the auxiliary function is defined in (21), we need to prove it satisfies the aforementioned conditions. Apparently, $Q(c, c) = O_{cij}(c)$. Hence, the left we need to prove is $Q(c, c^t_{ij}) \leq O_{cij}(c)$. To prove this, we compared $Q(c, c^t_{ij})$ shown in (21) with the Taylor expansion of O_{cij} near to c^t_{ij}

$$O_{c_{ij}} = O_{c_{ij}} + O'_{c_{ij}}(c_{ij} - c_{ij}^t) + \frac{1}{2} O''_{c_{ij}}(c_{ij} - c_{ij}^t)^2 \tag{22}$$

where O'_{cij} and O''_{cij} are the first and second order partial derivatives relevant to c_{ij}. Note that

$$
\begin{aligned}
O'_{c_{ij}} &= \frac{\partial O}{\partial c_{ij}} = \left[\alpha \mathbf{YD} + (1-\alpha)\mathbf{AB} - \mathbf{CD^TD} - \mathbf{CB^TB} - \mathbf{C} - \lambda \mathbf{Ce_1e_1^T} + \lambda \mathbf{e_2e_1^T}\right]_{ij} \\
O''_{c_{ij}} &= \frac{\partial^2 O}{\partial (c_{ij})^2} = -\left[\mathbf{D^TD} + \mathbf{B^TB} + 1 + \lambda \mathbf{e_1e_1^T}\right]_{jj}
\end{aligned}
\tag{23}
$$

Using (23) to replace the relevant terms in (22), we can see that if $Q(c_{ij}, c^t_{ij}) \le O_{cij}(c_{ij})$, the following inequality must hold

$$
\begin{aligned}
-\frac{(\mathbf{CD^TD} + \mathbf{CB^TB} + \mathbf{C} + \lambda \mathbf{Ce_1e_1^T})_{ij}}{2c^t_{ij}} &\le \frac{1}{2} O''_{c_{ij}} \\
&= -\frac{1}{2}\left[\mathbf{D^TD} + \mathbf{B^TB} + 1 + \lambda \mathbf{e_1e_1^T}\right]_{jj}
\end{aligned}
\tag{24}
$$

Therefore, to show $Q(c_{ij}, c^t_{ij}) \le O_{cij}(c_{ij})$, it is equivalent to show

$$
(\mathbf{CD^TD} + \mathbf{CB^TB} + \lambda \mathbf{Ce_1e_1^T})_{ij} \ge c^t_{ij}\left[\mathbf{D^TD} + \mathbf{B^TB} + \lambda \mathbf{e_1e_1^T}\right]_{jj} \tag{25}
$$

Since c_{ij}, d_{ij} and b_{ij} are non-negative, we have

$$
\begin{aligned}
&(\mathbf{CD^TD} + \mathbf{CB^TB} + \lambda \mathbf{Ce_1e_1^T})_{ij} = \\
&\sum_l c^t_{il}(\mathbf{D^TD})_{lj} + \sum_l c^t_{il}(\mathbf{B^TB})_{lj} + \lambda \sum_l c^t_{il}(\mathbf{e_1e_1^T})_{lj} \\
&\qquad \ge c^t_{ij}(\mathbf{D^TD})_{jj} + c^t_{ij}(\mathbf{B^TB})_{jj} + \lambda c^t_{ij}(\mathbf{e_1e_1^T})_{jj}
\end{aligned}
\tag{26}
$$

Up to here, $Q(c, c^t_{ij}) \le O_{cij}(c)$ has been proved thus (18) is an auxiliary function for O_{cij}.

Next, we will define the auxiliary functions regarding to the updating rules (16) and (19). Similarly, let O_{dij} and O_{bij} be the partial of (8) relevant to d_{ij} and b_{ij} and $O_{dij}(d'_{ij})$

and $O_{bij}(b'_{ij})$ be the partial objective values when d_{ij} and b_{ij} equal to d'_{ij} and b'_{ij}, respectively. Since the updating rules for **D** and **B** are also element wise, it is sufficient to show that O_{dij} and O_{bij} are non-decreasing according to the updating rules (16) and (19). Let the following be the auxiliary functions regarding to O_{dij} and O_{bij}:

$$\begin{aligned} Q(d_{ij}, d_{ij}^t) &= O_{d_{ij}} + O'_{d_{ij}}(d_{ij} - d_{ij}^t) - \frac{(\mathbf{DC^TC + D})_{ij}}{2d_{ij}^t}(d_{ij} - d_{ij}^t)^2 \\ Q(b_{ij}, b_{ij}^t) &= O_{b_{ij}} + O'_{b_{ij}}(b_{ij} - b_{ij}^t) - \frac{(\mathbf{BC^TC + B})_{ij}}{2b_{ij}^t}(b_{ij} - b_{ij}^t)^2 \end{aligned} \tag{27}$$

Since the proof for the above functions to be auxiliary functions regarding d_{ij} and b_{ij} is similar to that for c_{ij}, we don't show the proof in detail due to the space limitation.

Having obtained the auxiliary functions for c_{ij}, d_{ij} and b_{ij}, now we can show the convergence of (8) using the updating rules (13), (16) and (19). According to (20), we have

$$c_{ij}^{t+1} = \arg\max_{c_{ij}} Q(c_{ij}, c_{ij}^t) = c_{ij}^t \frac{(\alpha\mathbf{YD} + (1-\alpha)\mathbf{AB} + \lambda\mathbf{e_2e_1^T})_{ij}}{(\mathbf{CD^TD} + \mathbf{CB^TB} + \lambda\mathbf{Ce_1e_1^T})_{ij}} \tag{28}$$

The above result is same to the updating rule (13). Since (21) is an auxiliary function, O_{cij} is non-decreasing when c_{ij} is updated according to (28) or (13). This is equivalent to say that O is non-decreasing when c_{ij} is updated according to (13) for c_{ij} is any element of **C**.

Similarly, we have

$$d_{ij}^{t+1} = \arg\max_{d_{ij}} Q(d_{ij}, d_{ij}^{t}) = d_{ij}^{t} \frac{(\alpha \mathbf{YC})_{ij}}{(\mathbf{DC}^{\mathrm{T}}\mathbf{C} + \mathbf{D})_{ij}}$$
$$b_{ij}^{t+1} = \arg\max_{b_{ij}} Q(b_{ij}, b_{ij}^{t}) = b_{ij}^{t} \frac{[(1-\alpha)\mathbf{AC}]_{ij}}{(\mathbf{BC}^{\mathrm{T}}\mathbf{C} + \mathbf{B})_{ij}} \tag{29}$$

The above results are same to the updating rules (16) and (19). Since (27) are auxiliary functions, O_{dij} and O_{bij} are non-decreasing when d_{ij} and b_{ij} are updated according to (16) and (19). This is equivalent to say that O is non-decreasing when d_{ij} and b_{ij} are updated according to (16) and (19), respectively. Since O is non-decreasing when **C**, **D**, and **B** are updated according to (13), (16) and (19), O will finally converge to the local optima.

4.2.5. The stopping criterion

As **C**, **D**, and **B** are iteratively updated, the objective value converges to the local optima asymptotically. Simultaneously, the variation of the three matrices, **C**, **D**, and **B** becomes less evident as the elements in each matrix are approximate to the magnitudes which lead the objective value to local optima. Thus, we may use the following stopping criterion to terminate the optimization process and MISAGA may obtain matrices **C**, **D**, and **B** that lead O to converge approximately.

$$\left|\mathbf{C}^{i} - \mathbf{C}^{i-1}\right|_{F} < \tau \tag{30}$$

where $\mathbf{C}^{i}$ stands for the sub-graph membership matrix after the ith iteration of updating, τ represents the predefined tolerance which the Frobenius norm of the difference of **C** between two iterations should satisfy. When τ is set to be a relatively small value, MISAGA may obtain a sub-graph membership matrix **C** which is very approximate to the optimal.

4.2.6. Summary of the algorithm

Having obtained the updating rules for **C**, **D**, and **B** and the stopping criterion for the optimization process, now we may describe the details of MISAGA. Based on the aforementioned description, the proposed algorithm can be summarized as the pseudo codes shown in Fig. 1. As it is seen in the figure, there are not many parameters that need to be input. After the parameter of weight justification α, maximum number of iteration *max_iteration*, tolerance for improvement τ, penalty factor λ and the number of sub-graphs k are determined, MISAGA will iteratively update the strength matrices **C**, **D**, and **B** till the variation of **C** between each two iterations is less than τ or the objective function converges to the local maxima. After the optimization process is terminated, MISAGA obtains the sub-graph membership matrix **C**, which contains the

```
Algorithm 1-MISAGA
Input:      Y, A, α, max_iteration, τ, λ, k
Output:     C, D, B
Randomly initialize C, D, B;
C=C./[(Ce1e1^T];

for count=1: max_iteration

        Fixing D and B
    update C according to (13);
        Fixing C
    update D according to (16);
    update B according to (19);

        if (|C^i - C^(i-1)|_F<τ)
                compute objective value according to (8);
                break;
        end if

end for

return C, D, B;
```

Fig. 1. Pseudo codes of MISAGA

optimal or approximately optimal membership between each vertex and k sub-graphs. Given **C**, MISAGA can identify the best sub-graph membership for each vertex in the attributed graph.

4.3. Experiment and analysis

To evaluate the effectiveness of MISAGA, we performed a number of experiments using both synthetic and real-world datasets. In this section, we describe the details of the data sets that we used. We also explain how experiments and what criteria we used to evaluate performance.

4.3.1. Experimental set-up and performance metrics

4.3.1.1. *Baselines for comparison*

Compared with other approaches for sub-graph detection, MISAGA has many desirable features and it would be interesting to find out how much these features make MISAGA better. For such a purpose, we have selected a number of popular algorithms for performance benchmarking. They include Affinity Propagation clustering (AP), Spectral clustering (SC), k-means clustering, Multi-Assignment Clustering (MAC), CESNA, Relational topic model (RTM) and ECDA. One of the reasons why they are selected is that they are relatively more popular algorithms and have all been used effectively to discover sub-graphs in various network graphs. Also, they are representatives of the three categories of topology-based, attribute-based and topology-and-attribute based algorithms respectively.

For example, AP and SC are sub-graph identification algorithms that mainly consider topological structures of a graph when performing graph clustering. These two algorithms can discover interesting sub-graphs with sizes that can be very different from those methods that perform graph clustering based on modularity optimization, e.g., CNM and BGLL. For our experiments, we used the SC that makes use of the

normalized cut in graph clustering. For sub-graph identification algorithm that are based on attribute values, *k*-means clustering and MAC are used as to cluster vertices based on the similarity of the attribute values associated with them.

For graph clustering algorithms that consider both graph topologies and attribute values, we used CESNA, RTM and ECDA. As most effective algorithms taking into consideration both graph topology and attribute values are model-based, we selected CESNA and RTM, which utilize generative models and topic models, respectively. While, ECDA performs its tasks using an evolutionary graph clustering algorithm. The three approaches are therefore very different even though they all take both topologies and attribute values into consideration.

For performance benchmarking, the algorithms above were not re-implemented. The source code or executables made available by the authors were used in our experiments. All experiments were conducted under the same environment which included a workstation with 4-core 3.4GHz CPU and 16GB RAM.

4.3.1.2. *Experimental set-up*

Other than ensuring that the algorithms we used for benchmarking purposes are tested with the original code written by the authors in the same computing environment, the parameters that are required for these algorithms to run are set in such a way that either the default settings as recommended by the authors are used or that they are tuned by trials to find the best settings.

Specifically, the AP, CESNA and ECDA algorithms do not require input parameters to be set by the users. For these algorithms, the default settings as recommended and implemented by the authors were used. For algorithms, including SC, *k*-means, MAC, and RTM, which require parameters to be manually input into the system, we tried as

many different settings as we can, to obtain the best results for performance benchmarking. For example, SC requires that the parameter of *sigma* and the number of clusters (k) be set by the users before it can run. To find a better set of parameters, we tried SC using different sigma and k settings from 1 to 10 and from 10 to 200 respectively. The settings that give the best performance of SC are recorded and presented in our performance analysis report below.

For MISAGA, we tried different settings of the parameter, λ, that is required for the algorithm to work. λ was set in our experiments to 10^{-1}, 10^{0}, 10 and 10^{2} and we found that MISAGA performed well when λ was set to 10^{-1} or 10^{0}. Thus, all the experimental results of MISAGA shown in this manuscript were obtained when λ was set to these values. As for the other parameters, we set α to 0.5, maximum iterations to 300, and τ to 1e-9. As for k, it is set to be the same as the other algorithms, k-means, MAC and RTM. All the algorithms, including MISAGA, were executed 10 times in each set of data to obtain average results.

4.3.1.3. *Data sets and descriptions*

For performance evaluations, we used both synthetic and real datasets with known ground truth. We used synthetic data to test the effectiveness and efficiency of different algorithms and we used the real data sets to test the robustness of the different algorithms. The data sets that we used are described below.

Ego-facebook. The *Ego-facebook* [74] data set is social network data that is constructed based on a number of sub-networks extracted from facebook.com. As such, interesting ground truth sub-graphs that represent social circles are known and can be used for evaluation. In this dataset, there are 4039 vertices each of which represents a facebook user. These vertices are connected by 88234 edges that represent the friendship between users and for each vertex, a total of 1283 attribute values, which represent the profile

of the user are associated with the vertex that represents the user.

Caltech. This is a set of social network data which is constructed based on the friendship relationship extracted from the California Institute of Technology. The social network users in *Caltech* can be segmented into different classes based on the house affiliation according to the college's dorm system [98]. There are 769 vertices representing 769 social network users, and 16656 edges representing the friendship between users. A total of 53 attribute values that represent the user profiles is associated with the vertex that represents the user.

Rice. This is a set of online social network data which is constructed based on the friendship relationship among students studying at the Rice University. There are altogether 4087 vertices representing students and 184828 edges representing the friendship between students represented as vertices. The student profile, which is made up of 74 attributes, is considered the set of attribute values that is associated with each vertex. The ground-truth communities of this data set have been identified based on dormitory residence [98].

Villa. This is another set of online social network data which was constructed based on the friendship relationship obtained from the University of Villanova. There are altogether 7772 vertices representing different users and 314989 edges representing the friendship between users. For each vertex, a set of 140 attribute values that represent the profile of the user is made associated with the vertex. The ground truth communities for this data set have been identified.

Krogan [56]. This is a set of protein-protein interaction (PPI) network data related to Saccharomyces cerevisiae. It is constructed based on known interactions between proteins. In *Krogan*, there are 2674 vertices representing proteins, 7075 edges

representing interactions between them and a set of 3064 attribute values representing GO terms [5], associated with each protein, that represents the properties and functions of each protein. Interesting sub-graphs in this set of data represent known protein complexes that can be found in a CYC2008 database [83]. In other words, the ground-truth sub-graphs of this set of PPI network is known.

DIP [105]. This is another set of PPI network data which are constructed based on the interactions between proteins. In this dataset, there are 4579 vertices representing proteins, 20845 edges representing interactions between proteins and 4237 attribute values associated with each protein representing biological properties and functions. Like the *Krogan* dataset, the ground-truth sub-graphs for this *DIP* data set are also known.

Syn1k. This is a set of synthetic data which is generated based on the rule that the probability of intra-sub-graph edges is higher than that of inter-sub-graph edges and that vertices in the same interesting sub-graph are more related to each other than those that are not. For this data set, we used 1000 vertices that are divided into 4 ground truth communities, 9900 edges, and 50 attribute values are made to associate with each vertex.

The above datasets are used to test the effectiveness of MISAGA and other algorithms. In addition, to test the scalability of MISAGA, we have generated several additional synthetic datasets ranging in size from 5,000 to 100,000 for our experiments.

4.3.1.4. *Performance metrics*

For the purpose of performance evaluation, we used three evaluation measures, including the Normalized Mutual Information (*NMI*), the Average Accuracy (*Acc*) [85] and the Mean Jaccard Similarity (*JS*) [110] measures. All of them are widely used for evaluating the validity of detected sub-graphs or graph clusters.

The *NMI* measures the overall accuracy of the matches between sub-graphs that are detected and those that are considered "ground truth". It is defined as

$$NMI = \frac{\sum_{C,C^*} \Pr(C_i, C_j^*) \log \frac{\Pr(C_i, C_j^*)}{\Pr(C_i)\Pr(C_j^*)}}{\max(-\sum_i \Pr(C_i)\log \Pr(C_i), -\sum_j \Pr(C_j^*)\log \Pr(C_j^*))} \tag{31}$$

where $\Pr(C_i, C_j^*)$ denotes the probability that vertices are in both the detected sub-graph i and the ground truth sub-graph j, and $\Pr(C_i)$ denotes the probability that a vertex is found to exist in detected sub-graph, i. The *NMI* considers both the size of each discovered sub-graphs and the ground-truth sub-graphs by computing a fraction ratio between the sub-graphs that are identified and those that are available in the ground-truth database. If the *NMI* measure is high, it means that the sub-graphs detected match well with the ground-truth sub-graphs.

Contrary to the *NMI*, the *Acc* measure evaluates individually detected sub-graphs. It is defined as

$$Acc = \sum_c \frac{|C_i|}{|C|} f(C_i, C^*) \tag{32}$$

where $|C_i|$ means the size of a detected sub-graph, and $f(.)$ stands for a particular mapping function between a detected sub-graph i and the ground truth. For our purpose, we define $f(.)$ to be the maximum overlap between detected sub-graph i and a ground-truth sub-graph. As a result, *Acc* evaluates the best matching of each detected sub-graph. A higher value of *Acc* therefore means that each detected sub-graph has a better match with the ground truth. The higher the *Acc* is, the more effective the algorithm can be considered to be. Given its definition, *Acc* emphasizes more on the discovered sub-

graphs when evaluating.

The Mean Jaccard Similarity (*JS*) measures the average degree of agreement between the detected and ground-truth sub-graphs that share the highest Jaccard Similarity with them. It is defined as

$$JS = \sum_{C_i^*} \frac{\max_{C_j \subset C} S(C_i^*, C_j)}{2|C^*|} + \sum_{C_j} \frac{\max_{C_i^* \subset C^*} S(C_i^*, C_j)}{2|C|} \tag{33}$$

where (i) $S(C_i^*, C_j)$ denotes the Jarccard Similarity between the ground-truth, C_i^*, and detected sub-graph, C_j, (ii) |C| and $|C^*|$ denote the number of discovered sub-graphs, and that of ground-truth clusters, respectively. *JS* is a harmonic mean of the bi-directional Jaccard Similarity measure computed between the discovered and the ground truth sub-graphs. It first computes the arithmetic mean of the sum of the best Jaccard Similarity between each sub-graph and one particular sub-graph in the ground-truth database. Then, it does the same between each ground-truth sub-graph and one discovered sub-graph. *JS* is then determined by aggregating the above two arithmetic means using the weights of 0.5 for each. *JS* can evaluate the quality of discovered sub-graphs by considering the proportion of overlap between them and the ground truth sub-graphs without being affected by the sizes of sub-graphs identified and those that are known to be ground-truth.

Even though they all can be considered as measures of the differences between detected and ground-truth sub-graphs, the *NMI*, *Acc* and *JS*, do not measure exactly the same aspects. In fact, what they measure can be considered as complementary. It is for this reason that all three measures are adopted as part of our evaluation criteria.

Besides directly comparing the performance obtained by MISAGA and other algorithms in our experiments, we also carried out some statistical tests to determine whether the performance obtained by MISAGA is significantly better than that obtained by other baselines. For this purpose, we used the single-sided *z*-test to determine whether MISAGA is significantly better than other baselines at the 95% confidence level.

4.3.2. Experimental results using synthetics data

4.3.2.1. *Performance on identifying sub-graphs*

For performance evaluation, we used a set of synthetic graph data containing 1000 vertices, 9900 edges and 50 attributes, to test the effectiveness of all different algorithms in discovering interesting sub-graphs. The community structure of the synthetic dataset is shown in Fig. 2. As mentioned above, the synthetic data are generated by assuming that the probability of vertices within the same sub-graph to be connected with other vertices to be higher than that of the probability between sub-graphs. For the purpose of our experiment, the data set *Syn1k* was generated by setting the probability of intra-

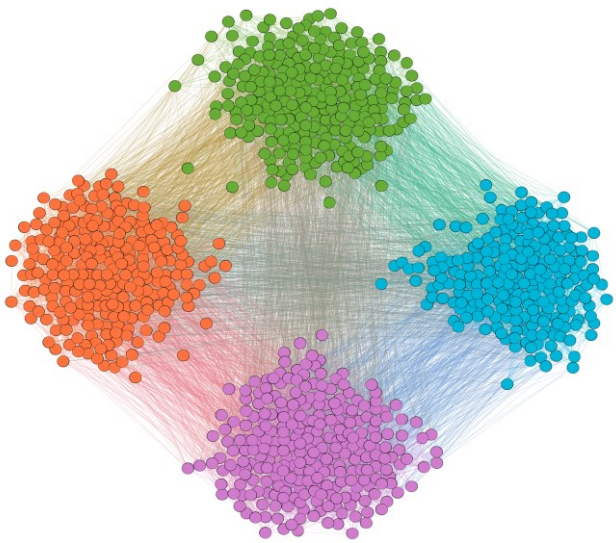

Fig. 2. Four ground truth sub-graphs of Syn1k. Vertices of the four sub-graphs are painted with different colors.

TABLE 1 *NMI*, *ACC* AND *JS* IN SYN1K

			Syn1k
Approach	*NMI*	*Acc*	*JS*
AP	0.152*	0.747*	0.028*
SC	0.232*	0.528*	0.329*
k-means	0.691*	0.835*	0.209*
MAC	0.745*	0.926	0.86
CESNA	0.792*	0.845*	0.748*
RTM	0.797*	0.797*	0.717*
ECDA	0.272*	0.466*	0.493*
MISAGA	**0.995**	**0.999**	**0.998**

The symbol * means the experimental performance obtained by MISAGA is significantly better than that obtained by a baseline algorithm at the 95% confidence level. The statistical test used in the experiment is single-sided *z*-test.

sub-graph connections to be 0.05 and the probability of inter-sub-graph connections to be 0.01.

The performance of MISAGA and other algorithms on the synthetic data set *Syn1k* with respect to *NMI*, *Acc* and *JS* is given in Table 1. As the table shows, MISAGA performs relatively better than other algorithms. When *NMI* measure is considered, the *NMI* obtained by MISAGA is better than RTM, CESNA, MAC, *k*-means, ECDA, SC and AP by 25%, 26%, 34%, 44%, 266%, 329% and 547%, respectively. When *Acc* is considered, MISAGA outperforms MAC, CESNA, *k*-means, RTM, AP, SC and ECDA by 8%, 18%, 20%, 25%, 34%, 68%, and 114%, respectively. When comparing performance using *JS*, MISAGA is better than MAC, CESNA, RTM, ECDA, SC, *k*-means and AP by 16%, 33%, 39%, 102%, 203%, 378% and 3464%, respectively. With the *z*-test, MISAGA is found to be better than any other baselines at a 95% confidence level when their discovered sub-graphs are evaluated by *NMI*. When evaluated by *Acc* and *JS*, the performance obtained by MISAGA is significantly better than all the baselines, except with MAC. These experimental results show that MISAGA can be very effective with the discovering of interesting sub-graphs.

4.3.2.2. *Scalability test*

In order to find out how MISAGA can scale up when data set size increases, a series of synthetic data of sizes ranging from 5000 to 100,000 were generated using the same probabilities of 0.05 and 0.01 for intra- and inter-sub-graph vertex connections as is with *Syn1k*. Given these generated data, the scalability of MISAGA was studied in a number of experiments involving different data sets. The results obtained were compared with those obtained with CESNA, RTM, and SC. As MISAGA and these algorithms are iterative in nature, the comparison is made based on the average execution time of each iteration. The results are shown in Fig. 3.

The results show that MISAGA scales up well when compared with CESNA, RTM, and SC. Even with the data sets containing as many as 100,000 vertices, MISAGA was able to complete each iteration in the optimization process in around 1 second and this is slightly faster than CESNA. However, when comparing the number of iterations that is required for the two algorithms to complete the sub-graph discovery tasks, it should be noted that CESNA needed at least 300 iterations whereas MISAGA converges much below 300. Given this to be the case, MISAGA is more computationally efficient.

When compared with RTM and SC, the computation time used by them was much more

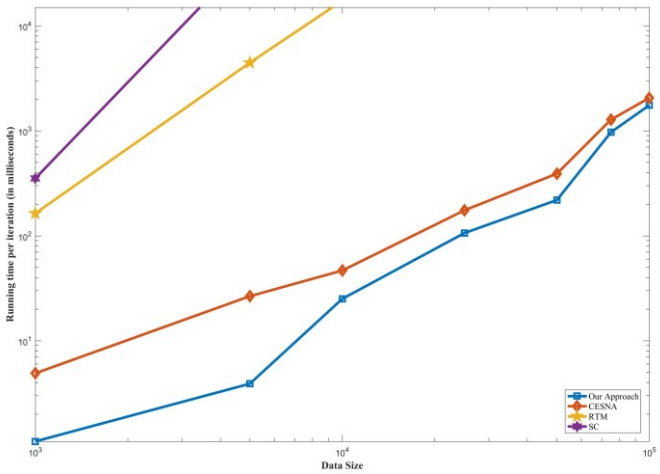

Fig. 3. Scalability comparison between MISAGA and other algorithms

demanding than MISAGA. When the data set size was increased to 10,000, RTM and SC was already not being able to cope. The computation time required was intolerable.

4.3.2.3. *Sensitivity test of the parameter*

As described in Section 3.2, for MISAGA to performs its tasks, it requires the setting of a parameter α. The parameter is used to adjust the weight between edge density and attribute association for interesting sub-graph discovery. How the parameter may affect the performance of MISAGA can be investigated in several sensitivity tests using the data set *Syn1k*.

In our experiment, α was set to different values from 0 to 1, with an increment of 0.2, and MISAGA was used under these different settings to try to discover interesting sub-graphs. The performance was measured with *NMI*, *Acc* and *JS* and the results are shown in Fig. 4. It is seen that when α was set to 0, which means that only the attribute values are considered in the sub-graph detection process, and when it is set to 1, which means that only the edge structures are considered, the performance of MISAGA is affected negatively. When setting α to the value between 0.4 and 0.6, MISAGA obtains very good results with most data sets. Given these results, we set α to be 0.5 in all our experiments so that both attribute values and edge structures are considered equally

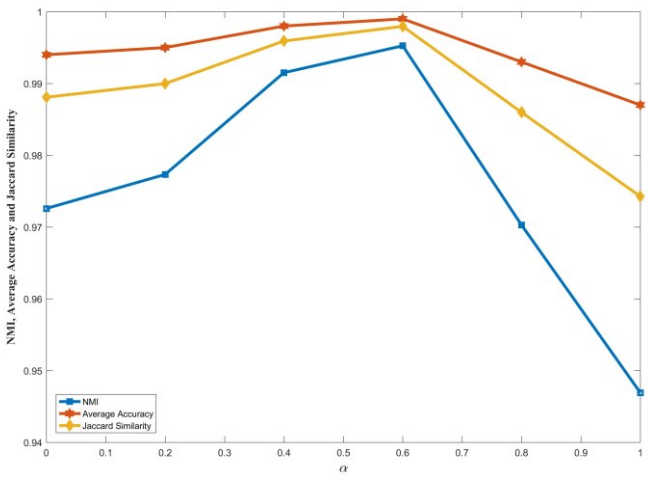

Fig. 4. Sensitivity test of MISAGA using different α

important by MISAGA.

In addition, to investigate how the settings of *k*, the total number of sub-graphs to be identified, may affect the performance of MISAGA, we used it to discover interesting sub-graphs in *Syn1k* by varying *k* from 2 to 20, with an increment of 2. The interesting sub-graphs identified were then evaluated by *NMI*, *Acc* and *JS* and the results are shown in Fig. 5. As shown in the figure, MISAGA performs the best according to *NMI*, *Acc*, and *JS* when *k* is configured appropriately (the best *k* for *Syn1k* is 4). Compared with the results of *Acc*, the *NMI* and *JS* as obtained by MISAGA seems to be more sensitive to the settings of *k*. This is because *NMI* and *JS* are two evaluation metrics which consider the extent of overall matching between discovered sub-graphs and the ground truth. An inappropriate setting of *k* might degrade the performance of MISAGA when its performance is measured by *NMI* and *JS*. As *Acc* mainly considers the best matching between a discovered sub-graph and a ground truth subgraph, MISAGA can obtain robust results using different settings of *k*.

With the results measured using *NMI*, *Acc* and *JS*, we can conclude that we can use MISAGA to discover interesting sub-graphs for real world applications using different values of *k*. However, we recommend that *k* can be either manually set between 2 and $n_V/2$ or done automatically by using the techniques described in [16], [72], and [99].

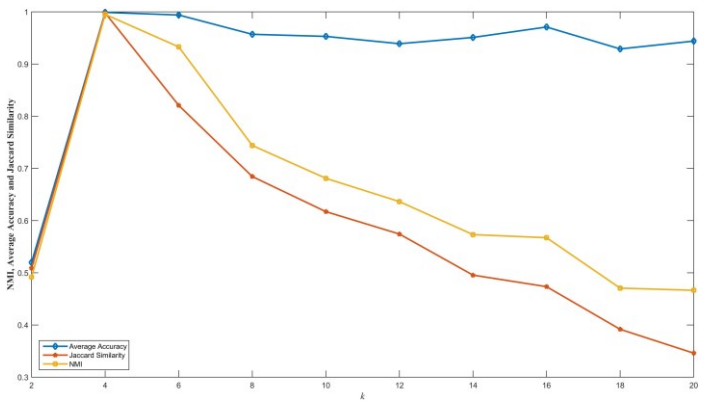

Fig. 5. Sensitivity test of MISAGA using different *k*

TABLE 2 EXPERIMENTAL RESULTS IN SOCIAL NETWORK DATA

Data set	*Caltech*			*Ego-facebook*			*Rice*			*Villa*		
Approach	*NMI*	*Acc*	*JS*	*NMI*	*Acc*	*JS*	*NMI*	*Acc*	*JS*	*NMI*	*Acc*	*JS*
AP	0.279*	0.458	0.066*	0.58*	0.416*	0.049*	0.139*	0.257*	0.019*	0.178*	0.441*	0.057*
SC	0.305*	0.375*	0.167*	0.569*	0.447*	0.107*	0.242*	**0.455**	0.066*	0.191*	0.492*	0.032*
k-means	0.176*	0.268*	0.131*	0.385*	0.276*	0.077*	0.055*	0.174*	0.044*	0.2*	0.381*	0.042*
MAC	0.106*	0.209*	0.081*	0.37*	0.257*	0.086*	0.024*	0.138*	0.056*	0.109*	0.325*	0.045*
CESNA	0.221*	0.384*	0.203	0.399*	0.384*	0.196	0.114*	0.222*	0.109*	0.336*	0.633	**0.285**
RTM	0.277*	0.23*	0.135*	0.592*	0.39*	0.161*	0.114*	0.195*	0.065*	0.375*	0.417*	0.102*
ECDA	0.156*	0.202*	0.242	0.353*	0.234*	0.105*	0.056*	0.147*	0.053*	0.245*	0.389*	0.169*
MISAGA	**0.453**	**0.487**	**0.257**	**0.675**	**0.582**	**0.203**	**0.368**	0.429	**0.222**	**0.495**	**0.69**	0.203

4.3.3. Experimental results in real applications

4.3.3.1. *Social community detection*

The social communities in a social network can be considered interesting sub-graphs in a social network graph. The identification of such social communities is important to social network analysis. For performance evaluation of MISAGA, we used four sets of real social network data, including *Ego-facebook, Caltech, Rice* and *Villa* as testing data sets. All these data sets have known ground-truth communities that have been verified in the previous work and for this reason, performance of the different algorithms can be more objectively compared.

The experimental results of *NMI*, *Acc* and *JS* obtained with these data sets are summarized in Table 2. As the table shows, MISAGA performs more robustly than other algorithms. For the data set *Caltech*, MISAGA outperforms SC, AP, and RTM by 49%, 62% and 64% when they are evaluated by *NMI*. When *Acc* is considered, MISAGA is better than AP, CESNA, and SC by 6%, 27%, and 30%, respectively. When evaluated by *JS*, MISAGA surpasses ECDA, CESNA, and SC by 6%, 27%, and 54%, respectively.

For the data set *Ego-facebook*, MISAGA surpasses, RTM by 14%, AP by 16% and SC by 19% in *NMI*. When *Acc* is considered, MISAGA outperforms SC by 30%, AP by

40%, and RTM by 49%, respectively. When evaluated by *JS*, MISAGA outperforms CESNA, RTM and SC by 4%, 26% and 90%, respectively.

For the data set *Rice*, MISAGA outperforms SC, AP and CESNA by 52%, 165% and 223% when they are evaluated by *NMI*. MISAGA also ranks the second best in *Acc*. When evaluated by *JS*, MISAGA outperforms CESNA, SC and RTM by 104%, 236% and 242% respectively.

For the data set *Villa*, MISAGA is better than RTM, CESNA and ECDA by 32%, 47% and 102% when *NMI* is considered. MISAGA ranks the second best evaluated by *JS*, following CESNA. When *Acc* measure is considered, MISAGA is better than CESNA, SC, and AP by 9%, 40% and 56% respectively.

In evaluating the performance of the different algorithms using the *z*-test, we can see that MISAGA outperforms most algorithms in most datasets. For examples, in the case of the *NMI* in all the datasets, MISAGA is statistically significantly better. When evaluated by *Acc*, MISAGA is also significantly better than most baselines in all the datasets, except AP in *Caltech*, SC in *Rice*, and CESNA in *Villa*. When evaluated by *JS*, MISAGA is statistically significantly better than any other baselines, except CESNA in *Caltech*, *Ego-facebook*, and *Villa*. Given such results obtained by the *z*-test, it is concluded that MISAGA is significantly more robust when used to discover meaningful sub-graphs in social network graphs. The experimental results also indicate that the interesting sub-graphs detected by MISAGA are consistently better matched with the ground-truth than the other algorithms.

4.3.3.2. *Structural modules detection in PPI networks*

Structural modules in biological networks, such as protein complexes in PPI network graphs can be considered interesting sub-graphs of the larger biological network graphs.

TABLE 3 EXPERIMENTAL RESULTS IN PPI NETWORK DATA

Data Set	*Krogan*			*DIP*		
Approach	*NMI*	*Acc*	*JS*	*NMI*	*Acc*	*JS*
AP	0.385*	0.187	0.168*	0.263*	**0.117**	0.138
SC	0.347*	0.129*	0.129*	0.174*	0.038*	0.038*
k-means	0.398	0.128*	0.12*	0.274	0.078*	0.086*
MAC	0.363*	0.114*	0.103*	0.3	0.062*	0.077*
CESNA	0.203*	0.025*	0.018*	0.122*	0.015*	0.011*
RTM	0.225*	0.054*	0.042*	0.157*	0.032*	0.029*
ECDA	0.416	0.142*	0.153*	0.303	0.058*	0.093*
MISAGA	**0.441**	**0.198**	**0.244**	**0.316**	0.113	**0.146**

To further test the effectiveness of MISAGA, we used two sets of PPI network data in our experiments. They included the *Krogan* and *DIP*. These data sets were chosen as the ground-truth sub-graphs, which correspond to known protein complexes, could be found. Performance data based on *NMI*, *Acc* and *JS* were obtained from the experiments. The results obtained with these two data sets are shown in Table 3.

As shown in the table, MISAGA obtains better performance than most of the other algorithms regarding of what performance measure are used. For the *Krogan* set, for example, MISAGA outperforms ECDA, *k*-means and AP by 6%, 11% and 15%, respectively, when evaluated by *NMI*. It surpasses AP, ECDA and SC by 6%, 39% and 53% when evaluated by *Acc*. When evaluated by *JS*, MISAGA outperforms AP, ECDA and SC by 45%, 59% and 89%, respectively.

For data set of *DIP*, MISAGA outperforms ECDA, MAC, and *k*-means by 4%, 5% and 15% when evaluated by *NMI*. When the detected sub-graphs are evaluated by *Acc*, MISAGA ranks the second best among all the algorithms. In the case that it is evaluated by *JS*, MISAGA performs better than other algorithms. MISAGA outperforms AP, ECDA, *k*-means by 6%, 57% and 70%, respectively. These results show that MISAGA is also very effective in identifying meaningful functional modules in protein-protein

interaction networks.

According to the *z*-test, the performance of MISAGA is also statistically significantly better than most baselines in the case of the biological graphs. *NMI*, *Acc*, and *JS* obtained by MISAGA are significantly better than those obtained by most baselines, except only five cases. Given the relatively robust performance of MISAGA with social network and PPI network data, we demonstrate that MISAGA can be a useful algorithm for mining interesting sub-graphs as clusters.

4.3.4. Convergence of the objective value

To find out if MISAGA can converge in a finite number of iterations, variations of the values of the objective functions were considered in different experiments. For experimentation, we randomly selected one execution of MISAGA when each set of real data is run and the variations of the values of the objective function are recorded as shown in Fig. 6. From it, it can be seen that the objective function of MISAGA can achieve approximate convergence within 200 iterations. As the improvement of each iteration become less evident, we may take the objective values after 200 iterations as the approximately convergent ones.

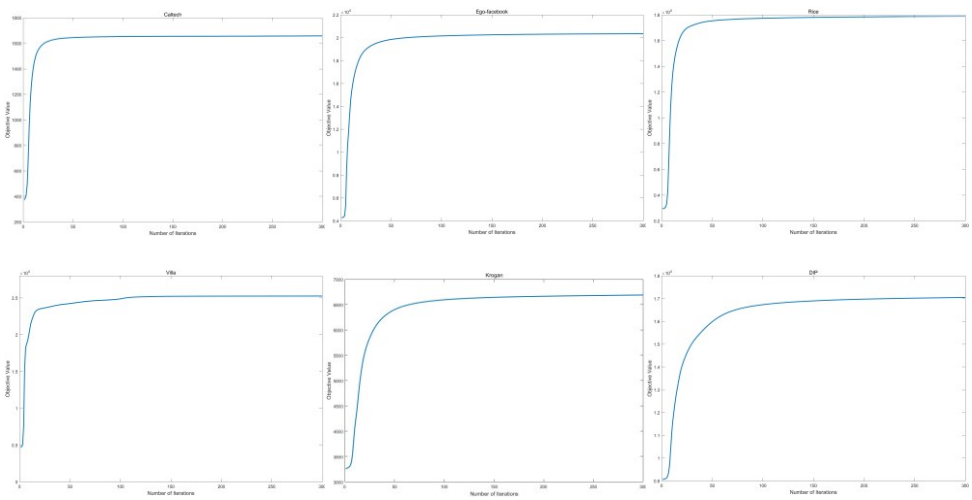

Fig. 6. Convergence of objective value in different real-world data

4.3.5. Case study of the detected sub-graphs

Besides evaluating the detected sub-graphs with the use of different objective measures such as *NMI, Acc,* and *JS*, we also investigated into the details of some of the interesting sub-graphs detected by MISAGA. In particular, we have looked into the details from the aspects of both edge structure and attribute values associated with these sub-graphs.

For example, in the dataset, *Ego-facebook*, one sub-graph identified by MISAGA completely matches with the community that is in the ground truth database. The structure of this community is shown in Fig. 7. Simultaneously, CESNA also identified this community, but the structure was not entirely identical (see the sub-graph inside the dashed circle in Fig. 7). Having checked each attribute value that is associated with each vertex in this community, we found that, even though two vertices in the sub-graph are connected, their respective attribute values are not exactly the same. Among the vertices in this sub-graph, only 15 out of over 1,200 attribute values that are associated

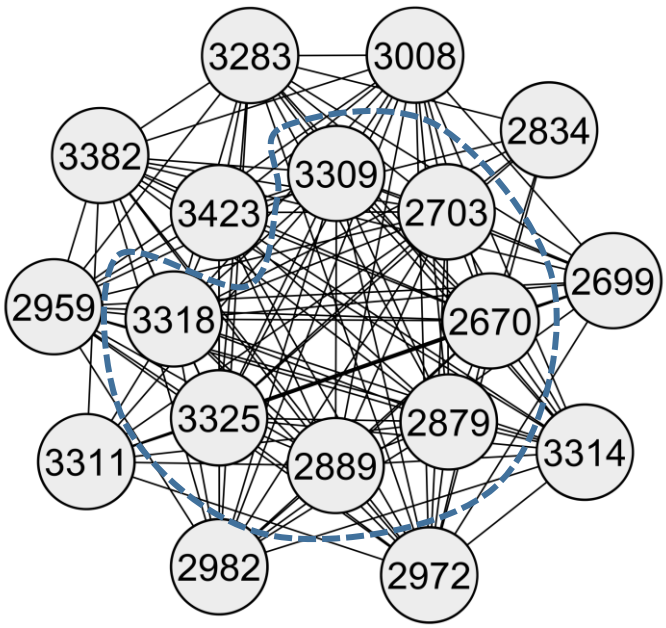

Fig. 7. The structure of a ground truth cluster in the data set of *Ego-facebook*. MISAGA identifies all the vertices successfully. The structure identified by CESNA is in the dashed circle.

TABLE 4 ASSOCIATED ATTRIBUTE VALUES IN THE DETECTED SOCIAL COMMUNITY

education;school;id;anonymized feature# 1040	education;school;id;anonymized feature 1040
education;school;id;anonymized feature 52	education;school;id;anonymized feature 52
education;school;id;anonymized feature 52	education;school;id;anonymized feature 52
education;school;id;anonymized feature 1040	education;year;id;anonymized feature 64
education;school;id;anonymized feature 1040	hometown;id;anonymized feature 935
education;year;id;anonymized feature 64	hometown;id;anonymized feature 935
education;school;id;anonymized feature 52	location;id;anonymized feature 128
location;id;anonymized feature 128	location;id;anonymized feature 128

#: The user profile data are processed as anonymized attributes to protect the privacy. In the table, each row contains a pair of associated attribute values. Associated attribute values might be different from each other.

with them. And these attribute values are not always the same. If a similarity measure is computed based on the attribute values, these vertices would probably not be considered to be in the same sub-graphs. Using MISAGA, however, we found 20 pairs of attribute values that are significantly associated with each other but not the same. For the purpose of discussion, we listed 8 pairs of them in Table 4.

Although the real user profiles are anonymized, we can infer that this social community might be related to a community of individuals that were friends of a school or university. As shown in the table, significant associations exist between different some attribute-value pairs. For example, “School 64” and “Hometown 935”, “School 1040” and “Hometown 935” between the “School” and “Hometown” attributes are found to be significantly associated. Such associations are expected as students from the same school may be friends and communicate more with each other if they are also from same district location. As another example, “School 1040” and “Year 64” being significantly associated may indicate that users in this community tend to communicate with others who entered the school in the same year. In addition, MISAGA has also discovered that there are significant interesting associations between users coming from the same school.

Instead of requiring that vertices in the same sub-graph share mostly the same attribute values, MISAGA is able to discover attribute values that are not the same but are associated with each other statistically significantly. The determination of sub-graph membership based on edge structure and such attribute association is the reason why MISAGA can better identify social communities.

4.4. Summary

In this section, a novel algorithm, that is called MISAGA, for mining interesting sub-graphs in attributed graphs is presented. MISAGA performs its tasks by considering both the edge structure and the attribute values that are associated with each vertex. By computing a degree of association for vertices that are statistically significantly associated with each other, MISAGA formulates and solves the problem of discovering interesting sub-graphs in an attributed graph as a constrained optimization problem. MISAGA has been tested with different sets of synthetic and real data. It is found MISAGA is effective in finding optimal or near-optimal solutions. For future work, how MISAGA can be enhanced to allow different vertices to belong to different sub-graph with different degrees of freedom will be interesting. In addition, it can also be enhanced to deal with attributed graphs with structures that change with time. The discovering of the dynamics of the changes will allow prediction of future structures to be made.

5. FSPGA-MINING CLUSTERS IN THE ATTRIBUTED GRAPH USING FUZZY OPTIMIZATION

5.1. Background

Sub-graphs in which vertices are cohesively inter-related are clusters or communities in the graph. These clusters are structural patterns hidden in the graph data. In this section, an algorithm for discovering fuzzy structural patterns, FSPGA, is proposed. FSPGA performs the task of clusters discovery as a fuzzy optimization problem which takes into consideration both graph topology and attribute values. FSPGA has been tested with both synthetic and real-world graph data sets and is found to be efficient and effective at detecting clusters in attributed graphs. FSPGA is a promising fuzzy algorithm for structural pattern detection in attributed graphs.

Recently, there have been several graph clustering algorithms proposed to detect clusters in graphs utilizing both edge structure and attribute information. Several examples have been listed in Section 2.

In addition, fuzzy pattern analysis, such as fuzzy clustering has been drawn much attention because the feature of "soft membership" that is possessed by the algorithms based on fuzzy techniques may lead one to detect more interesting sub-structures in different types of data. Besides of the classical fuzzy c-means algorithm [9], there are several algorithms based on the fuzzy c-means model, such as relational fuzzy c-means [43], fuzzy c-regression models [49], possibilistic fuzzy c-means models [84], and interval-based fuzzy model [92], which have been proposed for data clustering. And there are several fuzzy clustering algorithms proposed to solve specific clustering problems, such as motion detection [79] and linguistic analysis in web documents [22]. Among those proposed algorithms, FCAN [44] is the one that utilizes fuzzy techniques to detect clusters in complex network data. FCAN may detect clusters by segmenting a data matrix in which each element represents the strength of the relationship between

pairwise data points. The entries of the data matrix are obtained by adding the binary value and the degree of similarity representing the connection and attribute similarity between pairwise vertices, respectively. Though effective to some extent, FCAN may not truly identify the strengths of topology and attributes values that may determine the cluster arrangement within the clustering process.

Given the prevalent works of clustering in graph data and fuzzy clustering algorithms, we have the following findings that may motivate us to develop a more suitable algorithm. First, most of the graph clustering algorithms detect clusters based on topological properties only, or the attribute information is not fully utilized, just like the work presented in [44]. Second, most of the approaches cannot detect overlapping clusters, which might be more desirable in some graph data, e.g., communities in social networks are sometimes overlapping. Last but the most, currently, there are no effective fuzzy algorithms for clustering in attributed graphs.

To overcome the mentioned challenges, we propose FSPGA, an algorithm for discovering fuzzy structural patterns in the forms of clusters in attributed graphs. FSPGA performs its tasks by formulating the identification of clusters in attributed graphs as a constrained optimization problem that takes into the consideration edge structure and attribute. FSPGA may identify the optimal membership arrangement that is determined by both edge structure and attribute information between vertices and clusters. By adopting the fuzzy sets theory, FSPGA may detect overlapping clusters in the attributed graph.

For performance evaluation, FSPGA is tested with both synthetic and real data sets including social and biological network graphs. The experimental results are verified against known ground-truth data. It is found that FSPGA obtains a better performance in both efficiency and effectiveness, compared with state-of-the-art graph clustering

algorithms and fuzzy clustering algorithms. Given the performance, FCDAG is a very promising fuzzy based algorithm for overlapping clustering in attributed graph data.

5.2. FSPGA in details

5.2.1. Mathematical preliminaries

Given an attributed graph containing n_V vertices and n_E edges, in which each vertex is associated with a set of attribute values, FSPGA models the attributed graph as G = (V, E, Λ), which has been shown in Section 3.

Given the vertices and edges in G, we use an adjacency matrix **M** of dimensions, n_V by n_V, to represent the connections between vertices in G so that an entry, m_{ij}, in **M** has the value, 1, if v_i and v_j are connected and, 0, if they are not.

Besides the topological information, we also use another n_V-by-n_V matrix **A,** to represent the pairwise relationship in terms of attributes between vertices in G. Hence, each entry in **A**, say a_{ij}, can be obtained by any measure that may evaluate how similar or related the vertices v_i and v_j are, given the attribute values associated with them. Here we assume that a_{ij} should be nonnegative and a higher magnitude of it means v_i and v_j are more related, given the attribute values associated with the two vertices.

Given adjacency matrix **M** and pairwise relationship matrix **A**, we use the following augmented matrix to represent the mutual information between any pair of vertices in G

$$\mathbf{Y} = \begin{bmatrix} \alpha\mathbf{M} & \mathbf{0} \\ \mathbf{0} & (1-\alpha)\mathbf{A} \end{bmatrix} \tag{34}$$

where the parameter α is used to adjust the bias between edge structure and attribute similarity. The data matrix **Y** has the dimension of $2n_V$ by $2n_V$, the mutual information

between pairwise vertices are located in the diagonal blocks of **Y**, while entries in other blocks are all zero-valued. Utilizing **Y**, FSPGA may perform the task of clusters detection in G.

5.2.2. The objective function based clustering algorithm

FSPGA performs the task of community detection using **Y**. To find optimal cluster membership for the vertices in G that takes into the consideration edge structure and attribute, FSPGA is considering to use an objective function to evaluate the overall quality of detected clusters.

To formulate the objective that is adopted by FSPGA, we firstly introduce an auxiliary matrix having the dimension of $2n_V$-by-k, **X**, where k is the number of the clusters to seek. FSPGA uses **X** to represent strength in terms of structure and attributes that a vertex belongs to a cluster. Specifically, the first n_V-by-k entries are used for representing the structural strength that a vertex belongs to a cluster, and the last n_V-by-k entries are used for representing the strength in terms of attributes that a vertex belongs to a cluster. Let x_{ij} be an element in **X**. The value of x_{ij} indicates either the structural strength or that in terms of attributes that vertex i belongs to cluster j, according to the subscripts of the element. Given the properties of **X**, it can be used to represent the overall strength that each vertex belongs to a cluster as **X** uses different blocks to consider the strength regarding structure and attribute, respectively. The aggregation of such strength can be obtained if an appropriate method can be used. Then, we introduce the cluster membership matrix **C**, which has the dimension of n_V by k. Each element of **C**, say c_{ij}, indicates the strength of membership that vertex i belongs to cluster j. Apparently, a higher value of c_{ij} means vertex i belongs to cluster j more possibly.

Given **Y**, auxiliary matrix **X**, and cluster membership matrix **C**, we propose FSPGA to

formulate the cluster detection in the attributed graph as the following objective function to be optimized

$$\begin{aligned} &\text{maximize} \\ &O = tr(\mathbf{S}^{\mathbf{T}}\mathbf{YX}) - \frac{1}{2}\left[\left|\mathbf{C}\right|_F^2 + \left|\mathbf{X}\right|_F^2 + \left|\mathbf{XC}^{\mathbf{T}}\right|_F^2\right] \\ &\mathbf{S}^{\mathbf{T}} = \left[\mathbf{C}^{\mathbf{T}}, \mathbf{C}^{\mathbf{T}}\right] \\ &\textit{subject to } \mathbf{X} \geq 0,\ \mathbf{C} \geq 0, \mathbf{Ce}_1 = \mathbf{e}_2 \end{aligned} \tag{35}$$

where (i) $|\mathbf{C}|^2_F$, and $|\mathbf{X}|^2_F$ are the matrix Frobenius norms of **C** and **X**, which are used to smooth the variables in these matrices, (ii) $|\mathbf{XC}^{\mathrm{T}}|^2_F$ is the matrix Frobenius norm of the product of **X** and the transpose of **C**. (iii), $\mathbf{e}_1$ and $\mathbf{e}_2$ are k-by-1 and n_V-by-1 vectors, in which all elements are 1's. With the use of the proposed objective function, FSPGA can have the advantage that it can discover graph clusters by taking into consideration both edge structure and attribute information between vertices in the graph.

By introducing the auxiliary matrix **X**, the cluster membership that is identified by FSPGA takes into consideration both edge structure and attribute information between pairwise vertices in G. Let us take the first term in (35) to explain how FSPGA determines the cluster membership. Having noted that $tr(\mathbf{S}^{\mathbf{T}}\mathbf{YX})$ aggregate the strength of structure and attribute of all clusters, $tr(\mathbf{S}^{\mathbf{T}}\mathbf{YX})$ increases much only when **C** and **X** allocate those vertices with more intra-edges and more similar attributes into the same cluster. Hence, variations of **X** may lead the corresponding cluster membership to variate accordingly. From the above description, **X** considers the strength of both structure and attribute simultaneously, (35) can be optimized when appropriate **X** and **C** is found. In other words, the best cluster arrangement that is in **C** is identified when (35) achieves convergence. Under such an arrangement of clusters, vertices with more intra-edges and inter-related attribute values lean to the same cluster more. Since the existence of **X**, those vertices with either relatively fewer intra-edges or lower inter-

relation of attribute values may be assigned with lower magnitudes of strength. Only when elements in **C** and **X** are assigned with appropriate values, the objective function O can be maximized. Then, **C** forms the optimal cluster arrangement.

5.2.3. The iterative updating algorithm

The proposed objective function is a constrained quadratic function. Based on KKT condition for constrained optimization problems, we may find the corresponding rules to iteratively update the matrices **C**, and **X** to search the local optima.

5.2.3.1. *Updating rule for **C***

Let γ_{ij} and λ_i be the Lagrange multipliers for the constraints of $c_{ij} \geq 0$ and $\Sigma_j c_{ij} = 1$. The Lagrange function L for **C** is

$$L(\mathbf{C},\boldsymbol{\gamma}) = O - tr(\boldsymbol{\gamma}^{\mathbf{T}}\mathbf{C}) - \boldsymbol{\lambda}^T(\mathbf{C}\mathbf{e}_1 - \mathbf{e}_2) \tag{36}$$

where $\boldsymbol{\gamma} = [\gamma_{ij}]$ and $\boldsymbol{\lambda} = [\lambda_i]$ are Lagrange multipliers for the constraints of the non-negativity of **C** and the sum-to-1 of variables in each row of **C**. Based on the KKT condition for constrained optimization, we have

$$\begin{aligned}
&\frac{\partial L}{\partial \mathbf{C}} = \alpha\mathbf{M}\mathbf{X}_1 + (1-\alpha)\mathbf{A}\mathbf{X}_2 - \mathbf{C} - \mathbf{C}\mathbf{X}^{\mathbf{T}}\mathbf{X} - \boldsymbol{\gamma} - \boldsymbol{\lambda}\mathbf{e}_1^T = 0 \\
&\boldsymbol{\gamma} \circ \mathbf{C} = \mathbf{0} \\
&\boldsymbol{\gamma} \geq 0 \\
&\mathbf{C}\mathbf{e}_1 = \mathbf{e}_2
\end{aligned} \tag{37}$$

where (i) "$\circ$" means the Hadamard product of two matrices with the same dimension, (ii) $\mathbf{X}_1$ and $\mathbf{X}_2$ are two block matrices obtained by dividing **X** between row n_V and n_V+1. Based on (37), we have the following element wise equation system

$$
\begin{aligned}
&[\alpha \mathbf{MX_1} + (1-\alpha)\mathbf{AX_2}]_{ij} - (\mathbf{C} + \mathbf{CX^T X})_{ij} - \gamma_{ij} - \lambda_i = 0 \\
&\gamma_{ij} \circ c_{ij} = 0 \\
&\gamma_{ij} \geq 0 \\
&\sum_j c_{ij} = 1
\end{aligned}
\tag{38}
$$

Given the first equation in (38), we have

$$
[\alpha \mathbf{MX_1} + (1-\alpha)\mathbf{AX_2}]_{ij} - (\mathbf{C} + \mathbf{CX^T X})_{ij} - \lambda_i = \gamma_{ij} \tag{39}
$$

Using (39) to replace β_{ij} in the equation of Hadamard product, we have the iterative updating rule for **C**

$$
c_{ij} \leftarrow c_{ij} \frac{(\alpha \mathbf{MX_1} + (1-\alpha)\mathbf{AX_2})_{ij} - \lambda_i}{(\mathbf{CX^T X} + \mathbf{C})_{ij}} \tag{40}
$$

In the above equation, one more unknown, λ_i, needs to be determined for the updating of the variables in **C**. Given the constraint that the sum of each row of variables is one (see Equation (38)), we have

$$
\sum_j c_{ij} \frac{(\alpha \mathbf{MX_1} + (1-\alpha)\mathbf{AX_2})_{ij} - \lambda_i}{(\mathbf{CX^T X} + \mathbf{C})_{ij}} = 1 \tag{41}
$$

Given Equation (41), λ_i can, therefore, be solved as

$$
\lambda_i = \frac{\left(\sum_j c_{ij} \frac{(\alpha \mathbf{MX_1} + (1-\alpha)\mathbf{AX_2})_{ij}}{(\mathbf{CX^T X} + \mathbf{C})_{ij}} \right) - 1}{\sum_j \frac{c_{ij}}{(\mathbf{CX^T X} + \mathbf{C})_{ij}}} \tag{42}
$$

Using the value of λ_i to replace the corresponding variable in (40), the iterative updating

rule, which is under the fuzzy clustering framework for **C**, can be obtained. With such an updating rule, the sum of each row in **C** is constrained to be 1 within the optimization procedure. As a result, a vertex in G may belong to more than one cluster due to the considerations of fuzzy cluster boundaries.

5.2.3.2. *Updating rule for **X***

Let γ_{ij} be the Lagrange multipliers for the constraints $x_{ij} \geq 0$, hence the Lagrange function L for **X** is

$$L(\mathbf{X}, \boldsymbol{\gamma}) = O - tr(\boldsymbol{\gamma}^{\mathrm{T}} \mathbf{X}) \tag{43}$$

where $\boldsymbol{\gamma} = [\gamma_{ij}]$ is the matrix of Lagrange multipliers for the non-negativity of **X**. Based on the KKT condition, we have

$$\begin{aligned} &\left[\mathbf{YS} - \mathbf{XC}^{\mathrm{T}}\mathbf{C} - \mathbf{X}\right]_{ij} = \gamma_{ij} \\ &\gamma_{ij} \circ x_{ij} = 0 \\ &\gamma_{ij} \geq 0 \end{aligned} \tag{44}$$

Given the equation system (44), the element-wise updating rule for **X** can be derived

$$x_{ij} = x_{ij} \frac{(\mathbf{YS})_{ij}}{(\mathbf{XC}^{\mathrm{T}}\mathbf{C} + \mathbf{X})_{ij}} \tag{45}$$

5.2.4. Summary of the algorithm

Given the description from 5.2.1 to 5.2.3, FSPGA can be summarized as the pseudo codes shown in Fig. 8. Once the number of clusters k, the adjust parameter α, maximum number of iteration and the minimum tolerance, τ are determined, FSPGA will automatically search for the optimal matrix of cluster membership, **C** in a finite number of iterations. After FSPGA is stopped according to the terminal condition, the obtained **C** can be seen as the approximately optimal cluster arrangement.

5.2.5. Determining the cluster affiliation

Having obtained the cluster membership for each vertex, FSPGA needs to determine the cluster affiliation for the vertices. Here, FSPGA may determine whether vertex v_i belongs to cluster j according to the following inequality

$$c_{ij} \geq \frac{\beta\sqrt{k-1}+1}{k} \tag{46}$$

where k is the number of clusters and β is a positive real number that is used to determine the extent of overlapping between identified clusters in the attributed graph. Here, β is a global parameter which is used to determine if each vertex, say v_i, belongs to cluster j after the optimization process. In addition, it should be noted that β is used only for the case of vertices whose degrees of cluster membership are not the highest for that vertex and FSPGA can discover disjoint clusters in an attributed graph when β

Algorithm 2-FSPGA

Input:	**Y**, α, *max_iteration*, τ, k
Output:	**C**, **X**

```
Randomly initialize C, X;
C=C./(Ce1e1^T);

for count=1: max_iteration

    Fixing X
    update λ and C using (42) and (40);
    Fixing C
    Update X using (45);
        if (|C^i - C^(i-1)|_F<τ)
            compute objective value using (35);
            break;
        end if
end for
return C, X;
```

Fig. 8. Pseudo codes of FSPGA

is set to zero. Given this setting, it should be noted that it becomes more possible for more vertices to be assigned only to those clusters with the highest cluster membership and the extent of overlapping between detected clusters becomes smaller when β is set to a relatively high value. Hence, β can be adjusted according to the demand of overlapping in different attributed graph data and the variations of β won't change the number of clusters.

5.3. Experiment and analysis

In this section, we describe the details of the data sets that we used. We also explain how experiments and what criteria we used to evaluate performance.

5.3.1. Experimental set-up and evaluation metrics

5.3.1.1. *Baselines for comparison*

To show the desirable features of FSPGA, we selected a number of graph clustering algorithms to compare with FSPGA. These algorithms include Affinity Propagation clustering (AP), Spectral clustering (SC), CoDa, Fuzzy *c*-means clustering (FCM), improved Relational Fuzzy *c*-means clustering (iRFCM), CESNA, Relational topic model (RTM) and ECDA. Selecting these algorithms as baselines is because they are either the latest algorithms or classical ones and have all been used effectively to detect clusters in various network graphs. Specifically, AP, SC and CoDa may detect graph clusters that take different topological properties of network graph data. For our experiments, we used the SC that makes use of the *normalized cut* in graph clustering. FCM may detect graph clusters making use of information of similarity between pairwise vertices in G. Therefore, we used the information in Λ as the input that is used to compute the similarity between pairwise vertices for FCM. As iRFCM is a version of FCM that can be used to discover graph clusters, we tested it using the same data as FSPGA uses. Algorithms like CESNA, RTM and ECDA are ones taking into consideration both graph topologies and attribute values. RTM has been shown to be a

very effective topic-model based approach to segment relational data. CESNA performs graph clustering using a generative process that determines cluster membership of a vertex by computing an estimate of the joint probability based on structure and vertex attributes. ECDA performs its tasks using an evolutionary graph clustering algorithm.

For performance benchmarking, we used the source code or executables made available by the authors. All the experiments were conducted under the same environment which included a workstation with 4-core 3.4GHz CPU and 16GB RAM.

5.3.1.2. *Experimental set-up*

To ensure that the algorithms we used in the experiment may obtain a robust performance, we tested them using the parameters in such a way that either the default settings as recommended by the authors are used or that they are tuned by trials to find the best settings.

Specifically, the AP, Coda, and ECDA algorithms do not require input parameters to be set by the users. For these algorithms, the default settings as recommended and implemented by the authors were used. For algorithms, including SC, FCM, iRFCM, and RTM, which require parameters to be manually input into the system, we tried as many different settings as we can, to obtain the best results for performance benchmarking. For example, SC requires that the parameter of *sigma* to be set by the users before it can run. To find a better set of parameters, we tried SC using different sigma from 1 to 10. The settings that give the best performance of SC are recorded and presented in our performance analysis report below. As for the number of clusters, k, we set it for those algorithms that need k as a predefined parameter, including, SC, FCM, iRFCM, CESNA, and RTM, to be equal to the number of ground truth clusters that are used for benchmarking.

For FSPGA, we set β to 0 when FSPGA discovers structural patterns in those datasets whose ground-truth clusters are disjoint. We set β to 3 for all those datasets whose ground-truth clusters overlap with each other. As for the other parameters, we set α to 0.5, maximum number of iterations to 300. As for k, it is set to be the same as the other algorithms, which is equal to the number of ground-truth clusters in each of the datasets. All the algorithms, including FSPGA, were executed 10 times to obtain statistical averages for the performance measures.

5.3.1.3. *Data description and attribute similarity used by FSPGA*

For performance evaluations, we used both synthetic and real datasets with known ground truth. We used synthetic data to test the effectiveness and efficiency of different algorithms and we used the real-world data sets to test the robustness of the different algorithms regarding different applications. The real data sets that we used are mainly categorized into two classes, including social network graph data and biological network graph data.

The data sets *Twitter*, *Ego-facebook*, and *Googleplus* [74] are obtained from real social networking sites. The vertices, edges and attributes in these data sets represent users of the social networks, friendship between users and user profiles, respectively. The detail information on *Ego-facebook* dataset has been shown in Section 4.3.1.3. The *Twitter* data set is constructed based on a number of social circles extracted from twitter.com. For this data set, we have 2511 vertices, 37154 edges, and 9067 attribute values. *Googleplus* is another set of online social network data which was constructed based on the sub-networks from plus.google.com. There are 7856 vertices, 321268 edges, and 2024 attribute values in the data set. The ground truth social communities for this data set have been identified. There are 132, 191, and 91 ground truth clusters which are used for benchmarking the identified clusters from datasets *Twitter*, *Ego-facebook*, and *Googleplus*, respectively.

Krogan [56], *DIP* [105], and *BioGrid* [93] are three sets of biological data that are constructed based on known interactions between proteins related to *Saccharomyces cerevisiae*. In these three data sets, the vertices, edges, and attribute values represent the proteins, protein-protein interactions and GO terms [5], respectively. The detail information on datasets of *Krogan* and *DIP* has been shown in Section 4.3.1.3. As for *BioGrid* dataset, there are 5640 vertices, 59748 edges, and 4286 attribute values. These three data sets have the ground-truth data stored in CYC2008 database [83] and there are 200 ground-truth clusters. Compared with those social network graph data used, *Krogan*, *DIP* and *BioGrid*, are sparser. Using these two types of data allows us to find out how robust the algorithms are when used with different types of graphs.

Syn1k is a set of synthetic data which is generated based on the rule that the probability of intra-cluster edges is higher than that of inter-cluster edges and that vertices in the same cluster are more related to each other than those that are not. The detail information on *Syn1k* has been shown in Section 4.3.1.3. It should be noted that, the ground truth clusters of all the real data sets overlap with each other to some extent. Specifically, the overlapping rates between pairwise ground truth clusters in datasets *Twitter*, *Ego-facebook*, and *Googleplus* are 0.00193, 0.00113, and 0.01913, respectively. And that in *Krogan*, *DIP*, and *BioGrid*, it is 0.0004.

The above datasets are used to test the effectiveness of FSPGA and other algorithms. In addition, to test the scalability of FSPGA, we have generated several additional synthetic datasets ranging in size from 5,000 to 100,000 for our experiments.

To determine the strength between pairwise vertices in terms of attribute values that are used by FSPGA, we use the *degree of attribute association* method which can be obtained using (3), (4), and (5). Having obtained the degrees of interrelationship in terms of attribute values, we use them to construct **A** that is used by FSPGA.

TABLE 5 *NMI*, *ACC* AND *FARI* IN SYN1K

	Syn1k		
Approach	*NMI*	*Acc*	*FARI*
AP	0.152	0.747	0.01
CoDa	0.116	0.43	0.097
SC	0.232	0.528	0.277
FCM	0.732	0.871	0.674
iRFCM	0.718	0.739	0.677
CESNA	0.792	0.845	0.813
RTM	0.797	0.797	0.683
ECDA	0.272	0.466	0.203
FSPGA	**0.992**	**0.998**	**0.995**

5.3.1.4. *Evaluation metrics*

For performance evaluation, we are considering different evaluation measures which are widely used for evaluating graph clustering algorithms and fuzzy clustering algorithms. For measures used for validating graph clusters, we used the Normalized Mutual Information (*NMI*), and the Average Accuracy (*Acc*) [85]. There are a number of measures for fuzzy clustering validity, such as Beni Index [106], Earth Mover's Distance [6], and several fuzzy Rand-Index-based measures [7]. In our experiments, we selected Fuzzy Adjusted Rand Index (*FARI*) [7] for evaluating the graph clusters discovered by different algorithms.

5.3.2. Experimental results using synthetic data

5.3.2.1. *Evaluation of clustering quality*

For performance evaluation, we used a set of synthetic graph data containing 1000 vertices to test the effectiveness of all different algorithms. As mentioned above, the synthetic data are generated by assuming that the probability of vertices within the same cluster to be connected with other vertices to be higher than that of the probability between clusters. For our experiment, the data set *Syn1k* was generated by setting the probability of intra-cluster connections to be 0.05 and the probability of inter-cluster connections to be 0.01.

The performance of FSPGA and other algorithms on the synthetic dataset *Syn1k* with respect to *NMI*, *Acc* and *FARI* is given in Table 5. As the table shows, FSPGA performs better than other algorithms. No matter which of *NMI*, *Acc*, or *FARI* is considered, FSPGA may outperform all the compared baselines in dataset *Syn1k*. These experimental results show that FSPGA can be very effective with the discovering of clusters in the synthetic attributed graph.

5.3.2.2. *Scalability test*

To find out how FSPGA can scale up when data set size increases, a series of synthetic data of sizes ranging from 5000 to 100,000 were generated using the same probabilities of 0.05 and 0.01 for intra- and inter-cluster vertex connections as is with *Syn1k*. Given these generated data, the scalability of FSPGA was studied in a number of experiments involving different data sets. The results obtained were compared with those obtained with CESNA, RTM and iRFCM and AP. As FSPGA and these algorithms are iterative in nature, the comparison is made based on the average execution time of each iteration. The results are shown in Fig. 9.

The results show that FSPGA scales up well when compared with RTM and iRFCM and AP. Even with the data sets containing as many as 100,000 vertices, FSPGA was

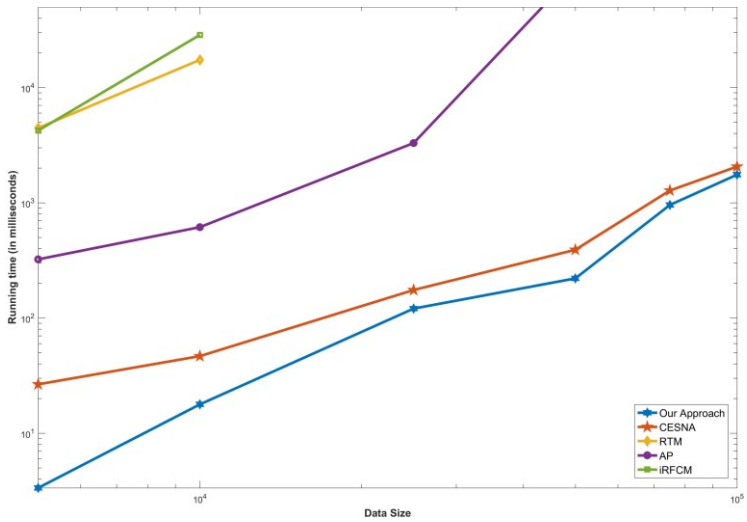

Fig. 9. Scalability test between different algorithms

able to complete each iteration in the optimization process in around 1 second and this is slightly faster than CESNA. However, when comparing the number of iterations that is required for the two algorithms to complete the cluster discovery tasks, it should be noted that CESNA needed at least 300 iterations whereas FSPGA converges much below 300. Given this to be the case, FSPGA is more computationally efficient.

When compared with AP, RTM, and iRFCM, the computational time used by them is much more than FSPGA did. It should be noted that we did not obtain the results of scalability test of RTM or iRFCM when the size of synthetic data is larger than 10,000 as they were crushed under that situation. And the computational time of AP is also intolerable when the data size is larger than 25,000.

5.3.2.3. *Sensitivity test of α*

As described in Section 5.2, for FSPGA to performs its tasks, it requires the setting of a parameter α. The parameter is used to adjust the bias between edge density and strength in terms of attribute within the process of cluster identification. How the parameter may affect the performance of FSPGA can be investigated in several sensitivity tests using the data set *Syn1k*.

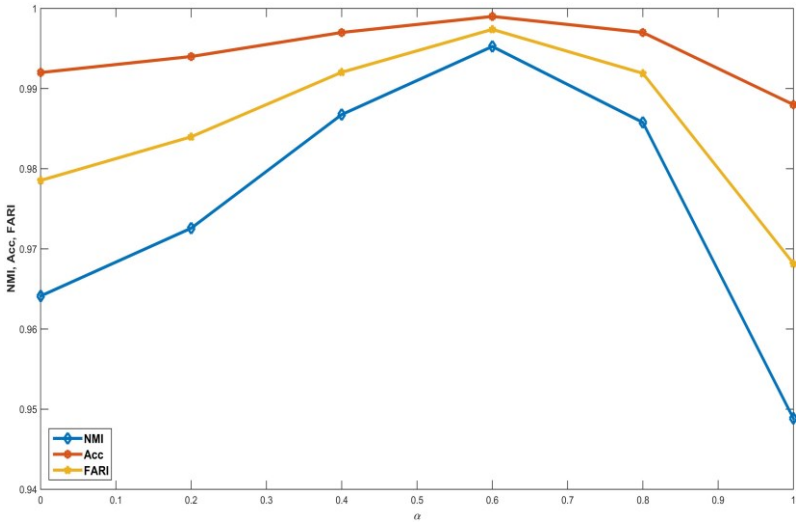

Fig. 10. Sensitivity test of α

TABLE 6 *NMI*, *ACC* AND *FARI* OBTAINED FROM SOCIAL NETWORK DATA

	Twitter			***Ego-facebook***			***Googleplus***		
Approach	*NMI*	*Acc*	*FARI*	*NMI*	*Acc*	*FARI*	*NMI*	*Acc*	*FARI*
AP	0.598^{2nd}	0.479^{3rd}	0.123	0.528^{2nd}	0.416	0.194^{1st}	0.355	0.273	0.095
CoDa	0.584^{3rd}	0.471	0.182^{3rd}	0.524^{3rd}	0.502^{3rd}	0.13^{3rd}	0.373	0.375^{3rd}	0.079
SC	0.493	0.305	0.094	0.52	0.447	0.126	0.33	0.296	0.081
FCM	0.08	0.09	0.016	0.28	0.208	0.056	0.128	0.181	0.031
iRFCM	0.535	0.37	0.172	0.315	0.282	0.074	0.266	0.318	0.054
CESNA	0.572	0.528^{1st}	0.169	0.483	0.623^{1st}	0.118	0.42^{2nd}	0.47^{2nd}	0.105^{3rd}
RTM	0.028	0.099	0.014	0.227	0.167	0.061	0.023	0.151	0.019
ECDA	0.529	0.385	0.184^{2nd}	0.322	0.234	0.099	0.395^{3rd}	0.341	0.122^{2nd}
FSPGA	**0.641^{1st}**	**0.513^{2nd}**	**0.241^{1st}**	**0.579^{1st}**	**0.588^{2nd}**	**0.17^{2nd}**	**0.489^{1st}**	**0.519^{1st}**	**0.149^{1st}**

In our experiment, α was set to different values from 0 to 1, with an increment of 0.2, and FSPGA was used under these different settings to detect clusters. The performance was measured with *NMI*, *Acc* and *FARI* and the results are shown in Fig. 10.

It is seen that when α was set to 0, which means that only the attribute values are considered, and when it is set to 1, which means that only the edge structure is considered, the performance of FSPGA is affected negatively. When setting α to the value between 0.4 and 0.6, FSPGA obtains very good results. Given these results, we set α to be 0.5 in all our experiments so that both attribute values and edge structures are considered equally important by FSPGA.

5.3.3. Experimental results in real data

5.3.3.1. *Results in social community detection*

Social communities are important structural patterns in social graphs. The identification of such communities is important to social network analysis. For performance evaluation of FSPGA, we used three sets of social network data, including *Twitter*, *Ego-facebook,* and *Googleplus*. All these data sets have known ground-truth communities that have been verified in previous work. Given the fact that the number of ground truth clusters is known, for those algorithms which need to set the number of clusters (k), we set it to be the number of known ground truth clusters in each dataset.

TABLE 7 *NMI*, *ACC* AND *FARI* OBTAINED FROM BIOLOGICAL NETWORK DATA

	Krogan			***DIP***			***BioGrid***		
Approach	*NMI*	*Acc*	*FARI*	*NMI*	*Acc*	*FARI*	*NMI*	*Acc*	*FARI*
AP	0.692[1st]	0.187[3rd]	0.11	0.688[2nd]	0.117[2nd]	0.098	0.109	0.016	0.003
CoDa	0.688[2nd]	0.199[2nd]	0.298[1st]	0.463	0.068[3rd]	0.045	0.299	0.035	0.017
SC	0.609	0.079	0.026	0.588	0.047	0.009	0.545[3rd]	0.032	0.087[3rd]
FCM	0.454	0.078	0.115	0.49	0.06	0.138[3rd]	0.444	0.048[2nd]	0.073
iRFCM	0.342	0.055	0.058	0.444	0.049	0.091	0.355	0.046[3rd]	0.045
CESNA	0.484	0.055	0.027	0.425	0.026	0.063	0.449	0.026	0.049
RTM	0.578	0.037	0.169	0.614[3rd]	0.025	0.184[2nd]	0.622[2nd]	0.021	0.194[2nd]
ECDA	0.631	0.142	0.229[2nd]	0.299	0.058	0.016	0.145	0.026	0.043
FSPGA	**0.677[3rd]**	**0.202[1st]**	**0.191[3rd]**	**0.712[1st]**	**0.129[1st]**	**0.267[1st]**	**0.755[1st]**	**0.125[1st]**	**0.372[1st]**

The experimental results of *NMI*, *Acc*, and *FARI* obtained with these datasets are summarized in Table 6. As the table shows, FSPGA performs more robustly than other algorithms. When the identified clusters are evaluated by *NMI*, FSPGA outperforms all the other algorithms in all the three social network datasets. When evaluated by *Acc*, FSPGA ranks the best in *Googleplus*, and second best in *Twitter*, and *Ego-facebook*, respectively. When the identified clusters are evaluated by *FARI*, FSPGA outperforms the other algorithms in the case of *Twitter*, and *Googleplus*, and ranks second best in *Ego-facebook*. In total, the above results obtained from social network data show that the social communities detected by FSPGA better match with the ground-truth when compared with the others.

5.3.3.2. *Functional modules detection in biological graph data*

Structural modules in biological networks, such as protein complexes in protein-protein interaction (PPI) network graphs can be considered as another important application of cluster detection in attributed graphs.

To further test the effectiveness of FSPGA, we used three sets of PPI network data in our experiments. They included the data sets *Krogan*, *DIP* and *BioGrid*. These data sets were chosen as the ground-truth, which correspond to known protein complexes, could

be found and some of the known protein complexes are overlapping. Performance data based on *NMI*, *Acc* and *FARI* were obtained from the experiments. The results obtained with these two data sets are shown in Table 7.

As shown in the table, FSPGA obtains better performance than all the other algorithms regardless of performance measures used. When the evaluation measure, *Acc* is considered, FSPGA outperforms all the baselines in all three datasets. When *NMI* is considered, FSPGA ranks the best in the case of *DIP* and *BioGrid*, and third with *Krogan*. When the discovered clusters are evaluated by *FARI*, FSPGA outperforms all other algorithms with *DIP* and *BioGrid*, and ranks third with *Krogan*.

As the objective function used by FSPGA considers pairwise relationship between any pair of vertices in terms of edge structure and attribute information, the relative weighting between how much each of these two factors should be considered can be adjusted dynamically during the optimization process. The fuzzy cluster membership matrix **C** obtained by FSPGA can find k clusters in which vertices share similar weighted structure with each other. This represents, in other words, the optimized weighted aggregation of intra-cluster connections and attribute relativity. This feature allows FSPGA to group unconnected but related vertices to be taken into consideration based on attribute relativity. Moreover, as FSPGA allows fuzzy cluster membership to be considered within the optimization process, thereby making it possible for FSPGA to find overlapping clusters as fuzzy structural patterns in an attributed graph. These features are the reasons why FSPGA can obtain a more robust performance with both overlapping and non-overlapping clustering.

5.3.4. Case study-overlapping rate versus clustering quality

To find out what impact the parameter β can have on the quality of clusters, FSPGA is tested with all real datasets, using β from 0.1 to 4, with a 0.1 increment. The clusters

obtained under different settings of β are evaluated using *NMI*, *Acc*, and *FARI*. The results are shown in Fig. 11 (a) - (c). Together with the overlapping rate of ground truth clusters of each real dataset, the impact of β on the quality of overlapping clustering were studied in detail.

The variations of *NMI*, *Acc*, and *FARI* are shown in Fig. 11 (a)-(c). As it is shown in the figures, the magnitudes of different clustering validity measures share similar variations. As the value of β becomes larger than a particular value, the clustering quality does not improve by much. In Fig. 11 (d), it should be noted that the extent of overlapping in all the datasets decreases and approximates to zero as β becomes larger and larger. Given these results, it should be noted that β needs to be adjusted for FSPGA to discover clusters with different extent of overlapping.

It is mentioned in 5.2.5 that β is used to constrain the number of vertices that can belong

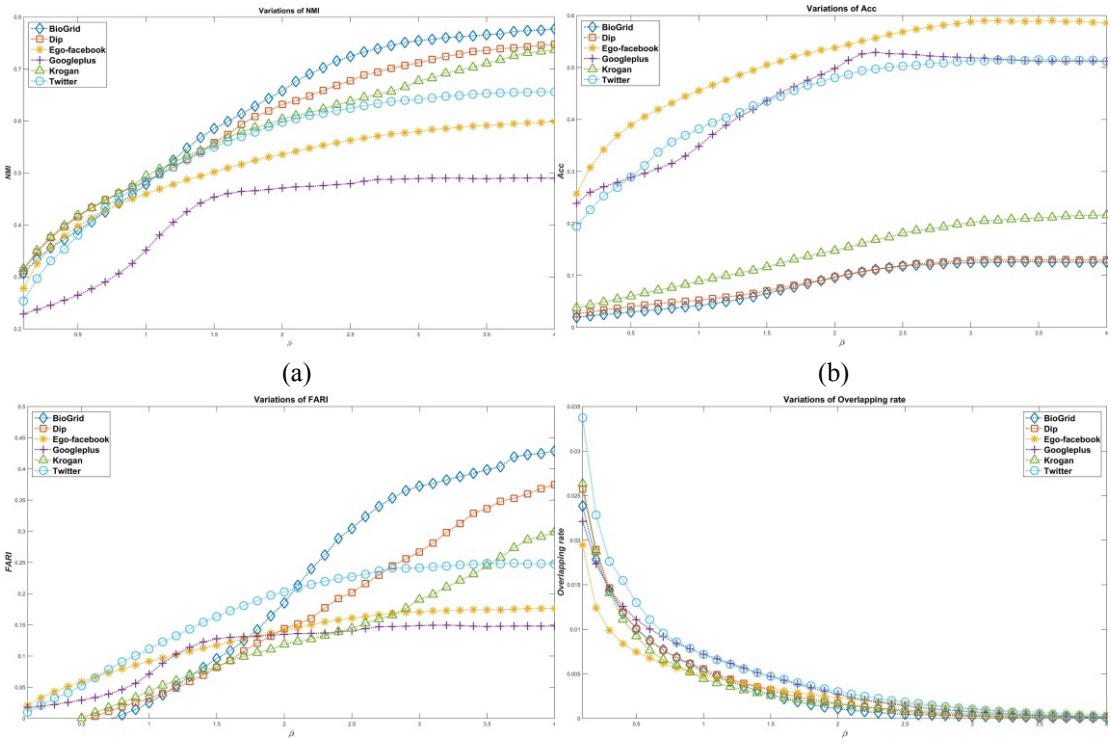

Fig. 11. *NMI, Acc, FARI,* and overlapping rate in social and biological datasets

to more than one cluster. As a result, each vertex in an attributed graph is probably assigned to the cluster with the greatest degree of cluster membership when β is set to be high enough. As a result, the clustering quality of FSPGA would be approximately the same as with crisp clustering. Given the fact that the overlapping rate of most datasets are relatively low, e.g., 0.00193 in *Twitter*, 0.00113 in *Ego-facebook*, and 0.0004 in biological datasets, the clustering quality is better when β is set higher, e.g., 2.5 to 3.5 in *Twitter*, *Ego-facebook*, and biological datasets. Using relatively large β, discovered clusters are mostly disjoint and the overall overlapping rate is therefore similar to that of the ground truth clusters (see Fig. 11 (d)).

However, when β is set between 2.5 and 3.5, it may degrade the quality of clusters discovered in *Googleplus*. This is because this data set has a relatively high overlapping rate. E.g., the *Acc* decreases when β is set larger than 2.5.

Given these characteristics of β, it is necessary for one to adjust the setting. However, FSPGA performs robustly when β is set between 2 and 3.5 and this is why β is set to 3 in our experiments.

5.4. Discussion

5.4.1. Comparisons between FSPGA and formal fuzzy clustering algorithms

With the above features, FSPGA can be considered different from such popular fuzzy clustering algorithms as the fuzzy *c*-means algorithm (FCM), and the relational fuzzy c-means algorithm (iRFCM). With the objective function that it uses, FSPGA can determine, dynamically, the degrees of cluster membership based on both the edge structure and attribute information of vertices. Compared with algorithms such as iRFCM which adjust global bias between **M** and **A**, FSPGA may identify meaningful clusters which other algorithms may miss.

Also, while existing fuzzy clustering algorithms minimize dissimilarity between data entities and cluster centers, FSPGA takes into consideration the edge structure and attribute relativity within each cluster. By so doing, FSPGA can identify clusters in which the weighted aggregation of intra-cluster connections and degrees of attribute relativity between any pair of vertices is optimized. These features are some of the reasons why FSPGA may perform better than other algorithms we used for comparison.

5.4.2. Computational complexity and space requirements of FSPGA

As the computational complexity of FSPGA is dependent mainly on the iterative updating of **C** and **X**, the complexity of the process of determining these matrices are considered here.

Let n_V and k be the number of vertices in the graph and the number of clusters in an attributed graph. It should be noted that $k << n_V$ in practice. According to Equation (40), for updating each element in **C**, say c_{ij}, it approximates the order of $O((2k+2)n_V)$. Hence updating all elements in **C** approximates the order of $O(k(2k+2)n_V^2)$. According to Equation (42), updating each element in $\boldsymbol{\lambda}$ follows the order of $O(k)$. This is because the computational components in the numerator and denominator are the same as those in (40). As a result, updating $\boldsymbol{\lambda}$ follows the order of $O(kn_V)$. According to Equation (45), the updating of each element in **X** approximates the order of $O((k+2)n_V)$. Hence updating all the elements in **X** follows the order of $O(2k(k+2)n_V^2)$.

Given the complexity of updating variables in **C** and **X**, FSPGA is an algorithm with complexity of $O(n^2)$. In other words, FSPGA is more efficient than the spectral-based clustering algorithms since their computational complexity follows the order of $O(n^3)$. In fact, as the augmented matrix **Y** is always very sparse, theoretically, FSPGA should run faster than those algorithms, such as Affinity Propagation (AP), that also have the complexity of $O(n^2)$. The scalability test shown in Fig. 9 also supports the analysis here.

Regarding the space requirement of FSPGA, it should be noted that FSPGA does not require much memory space when performing the task of discovering fuzzy structural patterns in the attributed graph. This is because FSPGA only stores those non-zero elements in the augmented matrix **Y**. For example, one synthetic data set used in our experiment contains 100,000 vertices, but there are about 30,000,000 elements in **Y** which are larger than zero. Compared with the full memory space for $100{,}000^2$, 30,000,000 is only 0.3% of the full space. Given the analysis here and the scalability test shown in Fig. 9, FSPGA can be used for discovering fuzzy structural patterns in large attributed graphs.

5.5. Summary

In this section, FSPGA, which is an algorithm for discovering fuzzy structural patterns in the form of clusters in the attributed graph, is proposed. Compared with prevalent algorithms that take different properties of an attributed graph, including topology, attribute, and both of the aforementioned, FSPGA may find an optimal arrangement of clusters for vertices in an attributed graph by formulating the task as a fuzzy constrained optimization problem. As the adoption of fuzzy set theory when determining the cluster membership, FSPGA can detect overlapping clusters, while most of the prevalent algorithms cannot. The experimental results presented in this paper show that FSPGA may perform robustly and efficiently in different types graph data, compared with the classical, latest graph clustering algorithms, and fuzzy clustering algorithms. In future, we will intend to further improve the efficiency of FSPGA and develop a version of FSPGA that may discover hierarchical structural patterns in attributed graphs.

6. EGCPI-AN EVOLUTIONARY ALGORITHM FOR IDENTIFYING CLUSTERS IN ATTRIBUTED GRAPHS

6.1. Background

As mentioned in Section 2, sometimes the task of mining clusters in some special network graphs needs treating particularly, e.g., protein complex identification in PPI network graphs. Such sub-graphs are always assumed to be connected components hidden in the attributed graph. Algorithms like MISAGA, and FSPGA need to combine with post-processing techniques for extracting connected components to ensure a better performance in such biological graphs. To develop more effective approaches that can be used in such attributed graphs, we propose two algorithms in Section 6 and 7. Given the experimental results that have been obtained, both two algorithms show their effectiveness in the task of sub-graph mining and outperform most state-of-the-art approaches. They are very promising methods for the given task.

To perform the task of discovering protein complexes in the PPI network graph, there have been several algorithms proposed. Among these methods, ones that consider both network topology and attribute information are few. For example, PCIFI [66] can identify protein complexes through searching sub-graphs in which proteins perform interdependent molecular functions. In [67], another algorithm may identify protein complexes in the PPI network graph utilizing gene expression data. In [117], an algorithm called GMFTP, is proposed to identify protein complexes based on measures of similarity between attribute values of proteins.

Based on most approaches that are proposed to identify protein complexes in PPI networks, we find that both topological and attribute information is very effective for identifying protein complexes, although there are not too many methods taking into the consideration both of the two types of information. We also find that most algorithms identify protein complexes by finding a number of clusters with some particular

properties that are optimized. Hence, sometimes the process of protein complex identification can be seen as an optimization problem. For such an optimization problem, we propose to tackle it with a novel approach based on an evolutionary algorithm called Evolutionary Graph Clustering for Protein Complex Identification (EGCPI). The advantages with the use of an evolutionary algorithm are that it does not have to work under linear constraints like those found in the typical numerical optimization problem. It can also discover multiple solutions and can be used to handle big data efficiently and effectively as it can be implemented in parallel.

Given a PPI network, EGCPI constructs an Attributed Graph by annotating attributes to each protein, based on the GO database which is constructed in Gene Ontology (GO) project [5]. To discover protein complexes, EGCPI assigns a weight to each edge in the graph, according to the *degree of topological similarity* which quantifies the proportion of common neighboring proteins shared by two interacting proteins. This transforms a PPI network graph G into a weighted graph denoted as wG.

Given a wG, EGCPI first finds a number of graph clusters in which the proteins are densely connected. It does so by optimizing an objective function that is defined in terms of the overall *degree of topological similarity* between connecting proteins in each cluster. After the identification of these dense graph clusters, EGCPI then makes use of a *degree of attribute homogeneity* measure to determine how similar the attribute values are between each pair of connecting proteins within each graph cluster. Based on the *degree of attribute homogeneity*, a breadth-first search strategy is then used to search for sub-graphs, within each graph cluster, that consist of proteins with similar attribute values. These sub-graphs are, therefore, relatively dense and their vertices share similar attribute values and they correspond well to the characteristics of many protein complexes in real life.

In order to evaluate the performance of EGCPI, we have tested it with different real data sets. The experimental results show that EGCPI performs well in terms of the number of accurately identified protein complexes that match with known protein complexes. We believe that EGCPI has great potential as an effective protein complex identifier.

6.2. EGCPI in details

Given a PPI network graph, EGCPI performs the task of protein complex identification in several steps. First, it models the PPI network graph using the notation (G) shown in Section 3. Second, a weighted Attributed Graph (wG) is then constructed by EGCPI. Given G, EGCPI determines a weight to each edge in the graph based on the *degree of topological similarity*. With all the weights for the edges determined, G is transformed into a wG. Given wG, an evolutionary algorithm is then used to identify graph clusters within which protein are densely connected by maximizing the overall *degree of topological similarity* in each cluster. Given the graph clusters, a breadth-first search strategy is used to search for subgraphs in each graph cluster based on the homogeneity of the attribute values associated with the connecting vertices. These subgraphs, whose vertices share similar attribute values and are relatively dense, are found to correspond well with protein complexes in real life.

6.2.1. Mathematical preliminaries

A PPI network can be represented as an attributed graph that contains n_V vertices that represent proteins and n_E edges that represent interactions between proteins. As we can obtain information about the attributes of each protein in a PPI network from the GO database, a set of attribute values can be considered as associating with each vertex in a PPI network graph so that this graph becomes an attributed graph. For our application, Λ contains three subsets, Λ_p, Λ_f, Λ_c corresponding to the attribute values of *biological processes*, *molecular functions,* and *cellular components*, respectively. *Biological*

processes is concerned with the biological objectives a protein is involved in. *Molecular functions* is concerned with the biochemical activities performed by a protein and *cellular components* is concerned with the location where a protein is most active in a cell. Thus, the PPI network graph can be represented using the notation in Section 3, G = (V, E, Λ), where V represents the set of vertices, E represents the set of edges and Λ represent the set of attribute values for proteins in G.

In order to explain how the above notations are used, let us assume that we are given a protein with Uniprot ID [102], Q08683, then the attributes in terms of the GO terms can be obtained. It should be noted that not all attribute information available about the protein is used in our experiments with the proposed algorithm because some of them may contain information about protein complex membership.

Other than avoiding the inclusion of these attribute values, it is noteworthy that the domains of the attributes are allowed to contain different values, i.e., it is possible that for any vertex in G, $|\Lambda^v_p| \neq |\Lambda^v_f| \neq |\Lambda^v_c|$. In addition, there is no requirement of EGCPI for any of the attributes of any vertex to have any value at all. Furthermore, there is also no requirement for the attribute value sets of two interacting proteins to have the same number of values for *degree of attribute homogeneity* to be determined.

6.2.2. Construction of wG

Given G, EGCPI makes use of a *degree of topological similarity* (σ) measure to weight each pair of interacting proteins according to how much they are connected. It is defined as

$$\sigma_{ij} = \frac{\left| e_{i+} \cap e_{j+} \right| + e_{ij}}{\left| e_{i+} \cup e_{j+} \right| - e_{ij}} \tag{47}$$

where e_{i+} is the set of vertices that are connected to v_i, e_{ij} equals to 1 if there is an interaction between v_i and v_j. σ evaluates the extent that a protein pair is connected to each other when considering the network topology.

The magnitude of σ ranges from 0 to 1. A higher value means that v_i and v_j are more highly connected. After a weight is determined for each edge in G, a weighted Attributed PPI network Graph (wG) is obtained. EGCPI then proceeds to try to find dense graph clusters using wG.

6.2.3. Evolutionary graph clustering

EGCPI identifies dense graph clusters using an evolutionary algorithm (EA). EAs have been shown to be very efficient at dealing with problems, such as *NP-Complete* problems that can otherwise be hard to tackle [14]. Recently, EAs have been used for graph clustering [45]. However, they have not been used for protein complex discovery in PPI networks. For EGCPI, an EA is used for the purpose of finding graph clusters that are more densely connected within a cluster than outside. This is because proteins in protein complexes are more densely connected in clusters as well.

Given a wG represented and encoded in a chromosome, the EA that EGCPI makes use of searches for an optimal solution based on a single-criterion objective function defined in terms of the weights in a wG. Like other EAs, EGCPI evolves an optimal graph clustering arrangement in several steps. To begin, a number of chromosomes is first initialized and their fitness values computed based on the weights of the graphs. Based on these fitness values, a particular number of coupled individuals are then selected for reproduction, which consists of both crossover and mutation. Chromosomes with lower fitness values will then be eliminated from the population after each generation of descendants are reproduced. The steps of selection and reproduction are then repeated.

6.2.3.1. *Gene representation*

EGCPI uses a *straight-forward representation* [50] to encode a graph clustering arrangement in a chromosome which has a length equal to the total number of vertices, n_V, in the graph. Assuming that S stands for the number of clusters in the wG and that S can be different in different chromosomes, then we can represent the case of the ith vertex in the wG being in the jth cluster as the ith gene containing the allele j.

6.2.3.2. *Initialization*

Since the *straight-forward representation* is used in EGCPI, the initial number of clusters, S must be determined before the initialization. Here S is a randomized parameter decided by the algorithm before each initialization of a chromosome. In the case of EGCPI, S is not to exceed a maximum value of $n_V/2$.

According to the typical approach of initializing a chromosome, a randomized cluster ID is assigned to each vertex. However, total randomness may result in long convergence time. To avoid the problem, EGCPI initializes a population of chromosomes with a different process that consists of the following steps: (i) S nodes is first selected randomly as the initial clusters, (ii) for each vertex v, with e connections (e>0), a cluster ID among S' is assigned to it, where $|S'|\leq S$ and S' is the set of cluster IDs to which vertices connected to v are possibly assigned; (iii) all the vertices without any connection are assigned to one of S clusters randomly. These three steps are iterated p times, where p is the size of the population. With this approach of initialization, p different ways of partitioning the vertices in wG into different numbers of clusters can be generated.

6.2.3.3. *Reproduction*

The reproduction process consists of both crossover and mutation. Traditionally, the uniform crossover operator which swaps alleles between two selected parent chromosomes is adopted by many EAs. However, EGCPI adopts a crossover operator

modified from the standard uniform crossover. In addition to the popular mutation operator that randomly changes the allele of a gene, EGCPI also makes use of a *Self-Variation* (*SV*) operator to introduce variation to a population of chromosomes.

As part of the crossover, instead of allowing all chromosomes to be selected for reproduction, EGCPI allows only a proportion of the chromosomes to be randomly selected and this proportion is set arbitrarily to 30%. This means that the best 30% of the chromosomes in the population can be candidates for reproduction only.

After selection, EGCPI performs crossover by first selecting one parent as a template. The cluster IDs of the other parent then replace all the alleles in the template which share the same gene positions with its cluster member according to the crossover rate. After crossover, each member of the selected cluster can then be selected for mutation based on the mutation rate. With these steps above, EGCPI generates a new descendant.

Although the fitness of the population is incrementally improved as new populations are continually generated, it takes a rather long time for the evolution process to converge. This is especially the case with large data set size. To tackle the problem, EGCPI adds an additional reproduction operator, called *Self-Variation* (*SV*). The main task of *SV* is to relocate a vertex to a cluster within which vertices share relatively higher weights with it, based on the current graph clustering arrangement. In other words, a vertex should belong to a cluster that shares connections of higher weights with it because connections of higher weights might lead to a higher probability of identifying protein complexes in the cluster. To achieve the expected goal, we use a two-dimension matrix to complete the stage of *SV*: Given all nodes in a network and all clusters, the matrix $VC_{[n_V][S]}$ is defined to represent the weight between a vertex and a cluster. For an element in VC, say vc_{ij}, it equals to zero if there is no connection between node *i* and other nodes in cluster *j*. Otherwise, it means some number of connections with a

magnitude of aggregated weight bridge node i and other nodes cluster j. Given $VC_{[n_V][S]}$, *Self-Variation* is completed like the follows: For a vertex v_i in cluster j, EGCPI firstly computes its expected weight $E(d_{vic})$ between v_i and each cluster, $E(d_{vic})$ is defined as:

$$E\left(d_{v_i c}\right)=\sum_{1}^{S}\frac{vc_{ij}\times vc_{ij}}{w_{v_i}} \tag{48}$$

where w_{vi} is the total weights of interactions whose one endpoint is v_i. Once $E(d_{vic})$ is obtained, EGCPI determines whether or not v_i should be relocated to a new cluster by

Algorithm 3-Evolutionary Clustering

```
Input: G
Output: A set of clusters C={c_i, 1≤i≤n_C}

generate σ for each interaction according to (47);
construct weighted Attributed PPI network Graph
(wG);
population initialization;
done = false;
while (done == false) {
        for (i = 0; i < maxdescendant; i ++) {
                crossover;
                Self-Variation;
                insertion of individual;
                elimination;
        }
        if(terminal-condition) {
                done = true;
        }
        else {
                if(re-initialization) {
                  population initialization
                  }
        }
}
return C;
```

Fig. 12. Evolutionary graph clustering

computing the largest difference of the real weight and the expected weight between v_i and each cluster, and this difference is defined as:

$$diff_{\max} = \max_{k} \left\lfloor vc_{ik} - E\left(d_{v_i c}\right) \right\rfloor \tag{49}$$

If $diff_{max} \neq j$, v_i will be moved to cluster $diff_{max}$, otherwise v_i will not be relocated. After the completion of *SV*, those clusters which possess relatively lower weight might be eliminated and the quality of clusters produced by crossover can be improved. Using the *SV* stage, EGCPI can find optimal dense clusters in wG in a less time.

6.2.3.4. *The fitness function*

Every time when EGCPI initializes the population or reproduces a new-birthed individual, the fitness value of each chromosome is computed and the highest fitness value is seen as the population fitness. In order to find partitions in which clusters possess more weight of intra-interactions but less that of inter-connections between clusters, we use a measure called *Independence of Cluster* (*IoC*) to evaluate a cluster that is partitioned by each individual in the population. And this measure is defined as

$$IoC_{C_i} = \frac{\sum_{v_j, v_k \in C_i} w_{jk}}{\sum_{v_j \in C_i} w_{jk}} \tag{50}$$

where c_i is the *i*th cluster in a partition, w_{jk} is the weight that is assigned to interaction e_{jk} in wG, the numerator and the denominator stand for the total weight of intra-interactions and that of all interactions connecting proteins in c_i. Once the *IoC* values of all clusters partitioned by an individual are obtained, we can evaluate the fitness of the individual by using the following objective function:

$$IoC_{wG} = \sum_{i=1}^{S} \frac{n_{V_i}}{n_V} IC_{C_i} \tag{51}$$

where n_{Vi} and n_V represent the total number of vertices in cluster *i* and wG, respectively. IoC_{wG} evaluates to what extent the independence is if a cluster is compared to another and it ranges from 0 to 1. Apparently, the higher IoC_{wG} is, the more independent from other ones a cluster is. Therefore, IoC_{wG} helps to diminish the interdependence between any two clusters.

Algorithm 4-Protein complex identification

```
Input: A set of clusters C
Output: A set of protein complexes PC
for each cluster c_i {
        generate θ for each interaction in c_i;
        for each vertex v_i {
        find θ_max;
        create a new protein complex r;
        create a new link list P_visiting;
        P_visiting = P_visiting ∪ v_i;
        P_visiting = P_visiting ∪ v_j;
        while(|P_visiting|>0){
            v_k = head of P_visiting;
            P_visiting- v_k;
            r = r ∪ v_k;
            search v_m: neighbors of v_k;
            if(θ_km≥λ×θ_max){
              P_visiting = P_visiting ∪ v_m;
              }
            }
          if(Ov_r≤OvMax){
             PC = PC ∪ r;
            }
        }
}
return PC;
```

Fig. 13. Protein complex identification

6.2.3.5. *Summary remarks*

Based on what has been illustrated from 6.2.3.1 to 6.2.3.4, the evolutionary algorithm is summarized as the pseudo codes shown in Fig. 12. When the phase of evolutionary clustering is finished, EGCPI obtains an individual containing optimal cluster arrangement for each protein in the wG. These clusters can be represented as $C=\{c_i, 1\leq i\leq n_C\}$, where n_C stands for the number of clusters.

6.2.4. Identifying protein complexes in found clusters

After using the above EA in wG, EGCPI obtains a set of clusters from the best individual in a population. In this stage, EGCPI performs a further extraction of subgraphs as protein complexes in each cluster. Rather than detecting protein complexes based on network topology, EGCPI identifies protein complexes by taking into consideration attribute homogeneity between pairwise proteins because proteins within each found cluster are already densely connected. Before searching protein complexes, EGCPI computes the *degree of attribute homogeneity* (θ) between each pair of connected proteins, using (1) in Section 3. θ determines how similar the attribute values are between each pair of connecting proteins within the cluster. It ranges between 0 and 1. A higher θ means there are more functional attributes performed by both of the connected proteins so that EGCPI tends to form protein complexes by searching such protein pairs in each graph cluster. After EGCPI uses the θ measure to weight all connected vertices, EGCPI uses a breadth-first search (BFS) method to form protein complexes in each cluster. First, it selects an interaction of a vertex with the highest θ, θ_{max}, and incorporates both of the two connected vertices v_i and v_j into a seed set for forming a protein complex; second, based on θ_{max}, EGCPI searches all the neighboring vertices and incorporates those which satisfy the minimum threshold of θ. In EGCPI, this threshold is defined as

TABLE 8 STATISTICS ON THE USED DATA SETS OF PPI NETWORKS

Data Set	n_V	n_E	n_A
Collins	1620	9064	2042
Gavin	1430	6531	2107
Krogan-Core	2674	7075	3064
DIP-Scere	4579	20845	4237
DIP-Hsapi	2434	3053	7031

n_V, the number of proteins; n_E, the number of interactions; n_A, the number of attributes.

$$r(seed : v_k) = \begin{cases} r \cup v_m \ if \ \theta_{km} \geq \lambda \times \theta_{\max} \\ r \cup \Phi \ otherwise \end{cases} \tag{52}$$

where v_k stands for a vertex in the seed set and v_m is a vertex connecting to v_k. In other words, only vertices sharing connections with θ which is higher than $\lambda \times \theta_{max}$ can be incorporated into the seed set. The searching in the second step will be terminated till there is no new vertex added to the seed set. When the above search in a cluster is finished, EGCPI forms a protein complex using the proteins in the seed set. EGCPI will stop forming protein complexes till it traverses all vertices in the cluster. As using the above search strategy may produce some protein complexes whose sizes are small, EGCPI discards those identified protein complexes including fewer than 3 proteins.

To reduce the redundancy of proteins in the identified protein complexes, EGCPI computes an overlapping score between an identified protein complex and protein complexes in the identified set. The overlapping score is defined as:

$$Ov_r = \max \frac{|r \cap PC_I|}{|r \cup PC_I|} \tag{53}$$

where r and PC_I stand for an identified protein complex after once of the search and any other protein complex that is in the identified set, respectively. Then EGCPI uses a threshold *OvMax* to exclude those identified protein complexes whose overlapping

scores are higher than the threshold. In order to explain this BFS method in detail, we give the pseudo codes in Fig. 13.

6.3. Experiment and analysis

For performance testing, EGCPI has been tested with five sets of real PPI network data. They include: (i) Collins [23], (ii) Gavin [37], (iii) Krogan-Core [56], (iv) DIP-Scere [105] and (v) DIP-Hsapi [105]. Datasets (i), (ii) and (iii), which can be collected from the BioGRID database [93], are concerned with *yeast Saccharomyces cerevisiae*. In our experiments, the data we used are collected from version 3.2.118 of the BioGRID database. Compared with Collins, Gavin, and Krogan-Core, DIP-Scere, which is also related to *Saccharomyces cerevisiae*, has a much larger data set size. Unlike the other data sets, DIP-Hsapi, is collected from human beings. Both the two DIP datasets are collected from the 2013 version of the DIP database [105]. The properties of these five datasets are shown in Table 8.

For all the five data sets, the protein attribute information required for the construction of the PPI network graphs were obtained from the January 2016 version of the GO database [24]. As mentioned above, the GO terms of the *cellular components* which may provide information about the protein complexes that a protein belongs to are not included in the experimental data sets.

To evaluate the performance of different protein complex identification algorithms, we compared the protein complexes identified with known protein complexes in *Saccharomyces cerevisiae* as contained in January 2016 version of the CYC2008 [83] and MIP/CYGD [75] [42] databases. There are 408 and 255 known protein complexes for *Saccharomyces cerevisiae* in the CYC2008 and MIP/CYGD databases respectively. Following the work completed in [78], we used the known protein complexes in these databases for performance evaluation. After removing protein complexes that are made

up of fewer than 3 proteins, we have obtained a total of 296 distinct protein complexes in the two databases for performance evaluation.

For the evaluation of protein complexes in the data set, DIP-Hsapi, we compared the protein complexes identified by different computational approaches with the known ones contained in the MIPS/CORUM [75] [88] database and there are altogether 1466 known protein complexes that are made up of three or more proteins in MIPS/CORUM.

6.3.1. Experiment set-up and performance evaluation

For performance evaluation, EGCPI were compared with different algorithms, including GMFTP, MCL, DPClus, IPCA, CFinder, COACH, SPICi and ClusterONE. We used these 9 algorithms to identify protein complexes in the five datasets described above. Algorithms like MCL, DPClus, IPCA, CFinder COACH, SPICi and ClusterONE identify protein complexes in PPI networks based only on network topologies. Attribute information that is made available are not considered by these algorithms. As algorithms like MCL, SPICi and ClusterONE can be used with weighted PPI network data, we also used them to identify protein complexes in the weighted PPI network data that are used by EGCPI. As GMFTP considers both network topology and functional attributes when identifying protein complexes, it is provided with exactly the same attribute information as we provided for EGCPI. For the above algorithms to perform their tasks, the settings of the parameters for each of them are given in Table 9.

When using EGCPI to find graph clusters, the population size is set to 100, and the

TABLE 9 PARAMETER SETTINGS OF DIFFERENT ALGORITHMS

Approach	Parameter	Approach	Parameter
ClusterONE	s=3, density=auto (default setting)	GMFTP	K=1000 (default setting)
MCL	inflation = 1.8 (default setting)	SPICi	minimum cluster size = 3
DPClus	CP_{in}=0.5, d_{in}=0.6 (default setting)	IPCA	S=3, P=2, T_{in}=0.4/0.9
CFinder	k=3	COACH	W=0.225 (default setting)
EGCPI	λ=0.7/0.8, *OvMax*=0.7/0.8/0.9		

TABLE 10 RESULTS OF *F-MEASURE* AND *MMR* OBTAINED FROM BIOGRID DATASETS

Data Set	Approach	#	Coverage	*f-measure*			*MMR*
				Precision	*Recall*	*f-measure*	
Collins	EGCPI	236	1160	0.67	0.54	0.6[1st]	0.29[2nd]
	GMFTP	203	1160	0.59	0.47	0.52[3rd]	0.28[3rd]
	ClusterONE	203	1293	0.54	0.45	0.49	0.25
	DPClus	203	1185	0.53	0.44	0.48	0.26
	MCL	282	1620	0.4	0.49	0.44	0.26
	SPICi	120	973	0.68	0.34	0.45	0.2
	IPCA	499	1160	0.48	0.66	0.55[2nd]	0.33[1st]
	COACH	245	1114	0.51	0.48	0.49	0.27
	CFinder	114	1160	0.7	0.32	0.44	0.2
Gavin	EGCPI	298	1150	0.66	0.56	0.6[1st]	0.27[1st]
	GMFTP	172	917	0.64	0.42	0.5[2nd]	0.22[3rd]
	ClusterONE	243	1268	0.4	0.37	0.38	0.19
	DPClus	217	1107	0.41	0.36	0.38	0.19
	MCL	177	1430	0.39	0.27	0.33	0.14
	SPICi	126	907	0.54	0.27	0.36	0.15
	IPCA	695	1124	0.36	0.61	0.46[3rd]	0.25[2nd]
	COACH	324	1052	0.43	0.46	0.44	0.22[3rd]
	CFinder	98	1124	0.56	0.2	0.29	0.11
Krogan-Core	EGCPI	526	1442	0.53	0.65	0.59[1st]	0.32[1st]
	GMFTP	299	1411	0.41	0.49	0.44	0.28[3rd]
	ClusterONE	242	1071	0.48	0.42	0.45[3rd]	0.23
	DPClus	497	1758	0.25	0.5	0.33	0.26
	MCL	514	2674	0.2	0.42	0.27	0.22
	SPICi	233	1239	0.38	0.36	0.37	0.19
	IPCA	701	1140	0.41	0.67	0.51[2nd]	0.3[2nd]
	COACH	349	1056	0.49	0.53	0.51[2nd]	0.27
	CFinder	115	1140	0.49	0.21	0.3	0.14

#: The number of protein complexes identified. Coverage: The number of distinct proteins in the identified protein complexes.

crossover rate is set to 0.6 for evolutionary clustering. We ran the EA for 30 generations before it was required to return the best partition of graph clusters. The reason why the maximum number of generations is set to 30 is because we found that the EA used in EGCPI could usually achieve the best results within around 30 generations. For λ, it

was set to 0.7 or 0.8. As for *OvMax*, it was set to 0.7, 0.8 or 0.9 to obtain a better performance.

For other algorithms, their parameters were set following the recommendations of the authors or modified as many times as possible to obtain a better performance. For example, different settings of T_{in} in IPCA have been proposed. In [103], it was set to 0.9 and in [68], it was set to 0.4 to obtain best results. For our experiments, we, therefore, used both these two settings to obtain a better performance. Another example is the parameter setting of CFinder. As there are no recommended settings for the size of the clique, we tried different values of k, from 3 to 50, and found that CFinder performed better when k was set to 3.

For the purpose of performance evaluation, we used two measures. One is the *f-measure* which can be taken as a measure that determines the overall accuracy of the identified protein complex. The *f-measure* can be defined as follows:

$$\begin{aligned} f-measure &= 2 \times \frac{precision \times recall}{precision + recall} \\ precision &= \frac{TP}{TP+FP}, \quad recall = \frac{TP}{TP+FN} \end{aligned} \tag{54}$$

The *f-measure* is determined by the value of *precision* and *recall*. *TP* is the number of identified protein complexes whose matching rates are equal to or larger than a particular threshold *O'*. In our experiments, we set the threshold to 0.2, which is recommended in [69]. *FP* is the number of identified protein complexes whose matching rates are less than the threshold *O'*. *FN* is the number of known protein complexes that are not matched by any identified protein complex. The other measure we used to determine the quality of identified protein complexes is the *Maximum Matching Rate* (*MMR*) [78]. Unlike the *f-measure*, it needs a predefined threshold to

TABLE 11 EXPERIMENTAL RESULTS OF *F-MEASURE* AND *MMR* OBTAINED FROM DIP DATASETS

Data Set	Approach	#	Coverage	*f-measure*			*MMR*
				Precision	*Recall*	*f-measure*	
DIP-Scere	EGCPI	441	2787	0.48	0.64	0.55[1st]	0.32[2nd]
	GMFTP	517	2509	0.28	0.60	0.38[3rd]	0.3[3rd]
	ClusterONE	335	1368	0.35	0.42	0.38[3rd]	0.19
	DPClus	856	2973	0.15	0.54	0.24	0.26
	MCL	691	4579	0.12	0.32	0.18	0.18
	SPICi	394	2055	0.24	0.39	0.3	0.18
	IPCA	1602	2142	0.23	0.8	0.36	0.37[1st]
	COACH	853	1952	0.27	0.7	0.39[2nd]	0.32[2nd]
	CFinder	192	2143	0.3	0.2	0.24	0.13
DIP-Hsapi	EGCPI	489	1241	0.37	0.13	0.2[1st]	0.05[1st]
	GMFTP	187	806	0.31	0.04	0.07	0.02
	ClusterONE	201	710	0.29	0.04	0.08	0.02
	DPClus	563	1644	0.19	0.09	0.12	0.05[1st]
	MCL	549	2434	0.17	0.07	0.1	0.04[2nd]
	SPICi	194	863	0.35	0.05	0.09	0.02
	IPCA	289	515	0.56	0.12	0.19[2nd]	0.05[1st]
	COACH	151	492	0.63	0.07	0.13[3rd]	0.03[3rd]
	CFinder	111	515	0.5	0.04	0.08	0.02

evaluate the identified protein complexes, *MMR* offers a natural way to measure how accurately the identified protein complexes can represent a benchmarking set. Based on the features of *f-measure* and *MMR*, these two evaluation criteria are complementary to each other.

6.3.2. Performance analysis

The experimental results of *f-measure* and *MMR* obtained by different algorithms have been summarized in Table 10 and 11. As the table shows, EGCPI obtains best *f-measure* in all the five data sets. Although EGCPI doesn't always obtain the best performance on *Precision* or *Recall*, but it makes a better compromise between the two measures so that the results of *f-measure* obtained by EGCPI are better than those done by other approaches. Given the results of *f-measure*, it is said the overall accuracy of protein

complexes identified by EGCPI is better than prevalent algorithms. When evaluated by the *MMR* measure, EGCPI also performed robustly in all the data sets. As the table shows, in the datasets of Gavin, Krogan-Core, and DIP-Hsapi, EGCPI ranks the first and it holds the second-best place in Collins and DIP-Scere. Given such results, it is concluded that the protein complexes identified by EGCPI may more accurately represent the known protein complexes in the benchmarking sets. As for the number of identified protein complexes and the coverage, EGCPI covered relatively more proteins when detecting protein complexes in each set of data, but it did not identify a large number of protein complexes. Together with the results of *f-measure* and *MMR*, it is seen that EGCPI is not an algorithm that obtains better experimental results by increasing the number of identified protein complexes.

To investigate whether other algorithms can obtain a competitive performance if they are used with the same weighted PPI network data, we compared the experimental performance obtained by those algorithms which can deal with weighted network data, including ClusterONE, MCL, and SPICi with that obtained by EGCPI. The results are summarized in Table 12. As the table shows, the performance of ClusterONE, MCL and SPICi is improved to some extent, but EGCPI still outperforms these three algorithms even when they use the weighted network data generated by the *degree of topological similarity*. Obtaining such results shows taking into consideration both topology and attribute makes EGCPI outperform those algorithms considering network topology only.

In total, EGCPI's performance on the task of protein complex identification is very promising. It obtains better results in both *MMR* and *f-measure* in most data sets. Therefore, EGCPI can perform better when it treats the task of protein complex identification as an optimization problem which takes into consideration both attribute information and topology of a PPI network.

6.3.3. The effect of parameter settings

As described before, there are two parameters in EGCPI, λ and *OvMax* determining the results of identified protein complexes. In order to investigate how these parameters, impact the results of protein complex identification, we executed EGCPI to identify protein complexes in the five data sets with λ and *OvMax* changing from 0.1 to 1.0, using a 0.1 increment. After collecting the identified protein complexes using different combinations of λ and *OvMax*, we evaluate them with *Precision*, *Recall*, *f-measure* and *MMR*. Here we take the variations of the above measures obtained in the data set Krogan-Core as an example (Fig. 14). As the surfs shown in Fig. 14 (a) and Fig. 14 (c), *Precision* and *f-measure* share an analogous trend when λ and *OvMax* change. Simply setting λ and *OvMax* to the values near to 0 or 1 may not obtain satisfying results. For example, EGCPI may obtain a relatively low *Precision* when λ is set to 0.2, no matter how to configure *OvMax*. When using a small λ, EGCPI may incorporate more proteins with lower *degree of attribute homogeneity* so that the protein complexes may not well match the known protein complexes. Although *Precision* and *f-measure* are relatively higher when λ and *OvMax* are set very near to 1, EGCPI cannot identify those protein complexes including more proteins so that some biological significance of the identified protein complexes is missing. Given such concerns, appropriate settings of λ and *OvMax* are essential to the experimental performance of EGCPI.

As Fig. 14 (b) and Fig. 14 (d) show, *Recall*, and *MMR* share the similar variations under different combinations of λ and *OvMax*. Using higher λ and *OvMax*, EGCPI may identify more protein complexes in the PPI network, so that it is possible for EGCPI to identify more protein complexes in the benchmarking set and higher *Recall* can be obtained. Since each identified protein complex including fewer proteins as λ and *OvMax* are set as higher values, its *MMR* is consequently larger than that of the identified protein complex including more proteins. Since we desire an approach that

can accurately identify protein complexes including relatively more proteins, in general, we recommend to let EGCPI perform the task of protein complex discovery when λ and *OvMax* are set between 0.6 and 0.9. EGCPI may obtain a robust performance when λ and *OvMax* are appropriately configured in that range. This is also the reason why we used the parameter settings for EGCPI which are shown in Table 9 in our experiments.

6.3.4. Complexity of EGCPI

To determine the efficiency of EGCPI, we analyze the complexity of EGCPI and recorded the execution time when it performed the task of protein complex identification. Here we mainly focus on the complexity of the evolutionary clustering as it is the dominant part of EGCPI. Unlike other algorithms, the complexity of EGCPI can be considered separately for initialization and reproduction. When EGCPI initializes a population of p individuals for a PPI network containing n_V vertices and n_E edges, for each chromosome which can be randomly initialized to contain S clusters, it

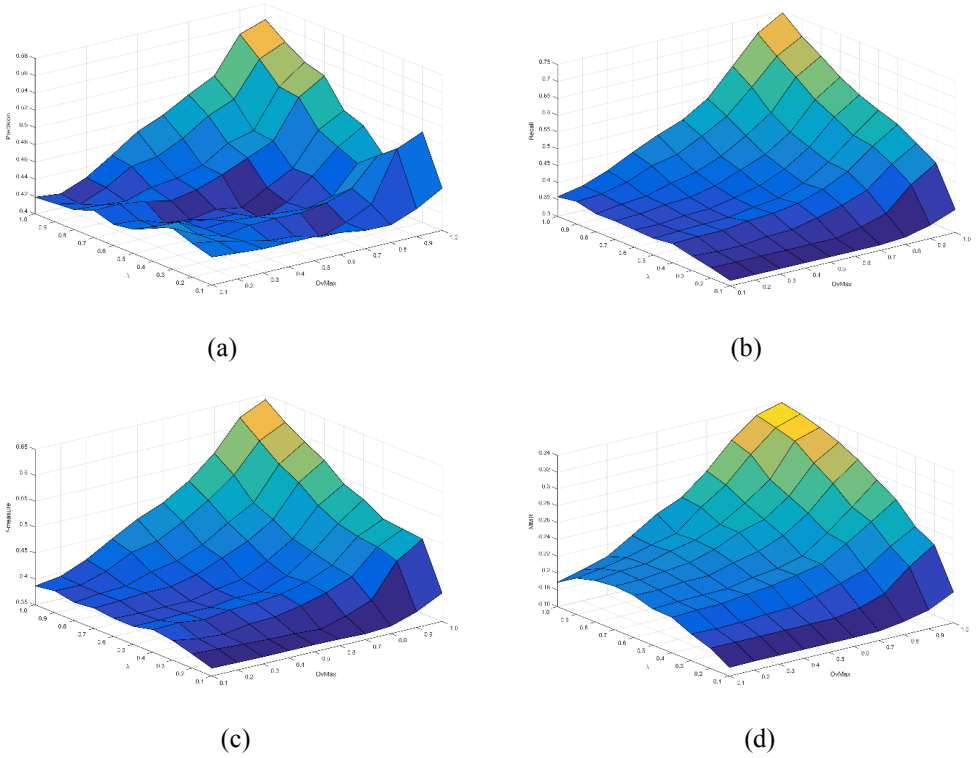

Fig. 14. Sensitivity test of EGCPI using different settings of λ and *OvMax*

performs its tasks requiring $O(2p(n_E+S))$, to construct the chromosomes and compute their fitness values. For reproduction, if d descendants are produced for each generation, and if the rate of crossover is r, EGCPI works under the complexity of $O(n_V r)$ and it takes $O(2n_E+n_V(4+r+S)+2S)$, for mutation and the computation of fitness. For the whole reproduction process, therefore, the complexity is $O(d(2n_E+n_V(4+r+S)+2S))$. If EGCPI takes g generations to achieve convergence, the complexity is of $O(2p(n_E+S)+dg(2n_E+n_V(4+r+S)+2S))$. Since $2p$ and dg are much smaller than n_E and n_V, we can assume that the two are equal to a constant, c. So, the complexity of EGCPI is approximately of $O(c(3n_E+Sn_V))$. But as reproduction progresses, the complexity should be much lower than this estimation since the operations for SV (i.e., $O(n_E+n_V(S+2))$) decrease tremendously as the evolutionary clustering goes on. In our experiment, we ran EGCPI on a workstation with 4 CPU (3.5GHz) and 16GB RAM. The total time consumption for evolutionary graph clustering in the largest dataset DIP-Scere was less than 3.5 seconds and that EGCPI reproduced individuals of each generation costing less than 0.15 second when we used the settings of EGCPI mentioned in Section 6.3.1. Given the complexity analysis and time-consumption recorded in the experiment, EGCPI can be seen as an efficient algorithm for protein complex identification.

6.3.5. Biological significance of identified protein complexes

Besides evaluating EGCPI by *f-measure* and *MMR*, we also investigated whether there was something biologically significant in the identified protein complexes.

To perform the investigation, we used GO::TermFinder [15] to make a functional enrichment analysis. Provided by SGD [25], GO::TermFinder is a web-based service that can be used for searching significant shared GO terms in the proteins of an identified protein complex. In our analysis, we set different thresholds of *p-value* from 1E-2 to 1E-15. In other words, those GO terms whose *p-values* are equal to or lower

TABLE 12 EXPERIMENTAL RESULTS USING WEIGHTED NETWORK DATA

Data Set	Approach	#	Coverage	*f-measure*			*MMR*
				Precision	*Recall*	*f-measure*	
Collins	EGCPI	236	1160	0.67	0.54	0.6	0.29
	ClusterONE	181	1207	0.59	0.44	0.5	0.26
	MCL	302	1620	0.4	0.52	0.46	0.28
	SPICi	94	819	0.79	0.31	0.44	0.18
Gavin	EGCPI	298	1150	0.66	0.56	0.6	0.27
	ClusterONE	156	950	0.6	0.37	0.46	0.2
	MCL	202	1430	0.44	0.36	0.39	0.17
	SPICi	83	532	0.75	0.25	0.37	0.13
Krogan-Core	EGCPI	526	1442	0.53	0.65	0.59	0.32
	ClusterONE	222	1007	0.5	0.42	0.46	0.25
	MCL	581	2674	0.2	0.48	0.29	0.24
	SPICi	65	435	0.86	0.22	0.35	0.13
DIP-Hsapi	EGCPI	489	1241	0.37	0.13	0.2	0.05
	ClusterONE	229	775	0.31	0.05	0.09	0.02
	MCL	644	2434	0.17	0.08	0.11	0.05
	SPICi	37	148	0.73	0.02	0.04	0.01
DIP-Scere	EGCPI	441	2787	0.48	0.64	0.55	0.32
	ClusterONE	243	919	0.52	0.44	0.48	0.21
	MCL	903	4579	0.13	0.46	0.2	0.23
	SPICi	44	235	0.7	0.11	0.19	0.07

In the experiment shown in this table, ClusterONE, MCL and SPICi identified protein complexes using weighted PPI network graphs that are generated based on the *degree of topological similarity* measure.

than the threshold may be identified as significant GO ones. Not all these protein complexes whose proteins share significant GO terms are known ones that can be found in databases such as MIPS/CYGD and CYC2008, but they can be considered as candidates of real protein complexes due to their statistical significance revealed by the functional enrichment analysis. Having obtained the *p-value* of each protein complex, we recorded the ratio that the identified protein complexes containing at least one GO term with the *p-value* lower than different thresholds in each GO category.

Besides analyzing the protein complexes identified by EGCPI, we also performed the

same *p-value* test on the protein complexes identified by GMFTP, MCL, IPCA and ClusterONE. GMFTP has been proved to be a very effective method which takes into consideration both network topology and functional attributes. MCL, IPCA, and ClusterONE are also proved to be effective methods considering network topology to identify protein complexes in the PPI network. Selecting the above approaches to compare with EGCPI is because all of them obtained robust performances in the five sets of data. Those approaches which did not perform robustly were not considered in the *p-value* test. The results of *p-value* test of EGCPI, GMFTP, MCL, IPCA and ClusterONE are presented in Table 13. As the table shows, the proportion of protein complexes with significant GO terms identified by EGCPI is higher than that of other algorithms, especially when the threshold of *p-value* is tightened (e.g., *p-value*<1E-15). This means EGCPI can identify more protein complexes with shared significant GO terms, compared with other approaches. Despite some of those identified protein complexes are not the known protein complexes currently, they have a higher possibility to be identified as real protein complexes through laboratory experiments in future. Based on the results of *p-value* test, it is seen that EGCPI is a promising approach to protein complex discovery.

Moreover, we also enumerate a number of matched protein complexes identified by EGCPI and select several protein complexes to make an analysis of both topology and GO information.

In Table 14, we enumerate 10 matched protein complexes identified by EGCPI. As the table shows, known protein complexes including more proteins like RSC complex, Mitochondrial ribosomal complex (small unit), Anaphase-promoting complex and PBAF complex can be successfully identified by EGCPI. Meanwhile, protein complexes including fewer sub-units such as Cytoplasmic exosome complex and

TABLE 13 P-VALUE TEST ON PROTEIN COMPLEXES IDENTIFIED BY DIFFERENT ALGORITHMS

Data Set	Approach	<1E-15	<1E-10	<1E-5	<1E-2
Collins	EGCPI	36.86%	60.59%	87.29%	94.91%
	GMFTP	31.53%	59.6%	85.71%	94.09%
	MCL	15.6%	28.72%	70.21%	83.69%
	IPCA	30.7%	50.3%	78.4%	90.2%
	ClusterONE	25.6%	48.8%	78.8%	93.1%
Gavin	EGCPI	42.62%	63.09%	86.91%	95.64%
	GMFTP	31.4%	51.74%	82.56%	92.44%
	MCL	22.59%	33.33%	64.97%	80.79%
	IPCA	14.9%	37.7%	72.9%	92.8%
	ClusterONE	20.9%	31.7%	60.5%	90.1%
Krogan-Core	EGCPI	32.13%	46.19%	74.52%	89.54%
	GMFTP	18.39%	35.45%	63.88%	69.23%
	MCL	8.37%	14.79%	43.17%	70.82%
	IPCA	14.6%	28.4%	67.6%	88.7%
	ClusterONE	21.9%	38%	68.2%	88.4%
DIP-Scere	EGCPI	33.56%	52.15%	82.54%	94.56%
	GMFTP	14.2%	26.26%	54.28%	78.99%
	MCL	8.17%	11.43%	32.56%	62.95%
	IPCA	3.9%	15.2%	58.1%	91.5%
	ClusterONE	15.5%	26.9%	60.6%	85.1%
DIP-Hsapi	EGCPI	22.49%	50.1%	93.66%	99.79%
	GMFTP	16.04%	39.04%	84.49%	97.33%
	MCL	9.29%	25.87%	71.22%	87.07%
	IPCA	1.4%	18%	92%	100%
	ClusterONE	9.5%	33.8%	84.6%	99%

Arp2/3 Complex can be detected by the proposed approach, too. Given such results, it is seen that EGCPI is effective for identifying protein complexes with different sizes.

In data set Krogan-Core, DASH complex was identified successfully by EGCPI. The structure stored in the CYC2008 database is shown in Fig. 15. It is noticed that proteins except P69850 and P69852 connect to each other densely. Due to this topological feature, P69850 and P69852 might be excluded from the protein complex by some algorithms based on network topology. For examples, CFinder, which is based on clique

TABLE 14 TEN MATCHED PROTEIN COMPLEXES IDENTIFIED BY EGCPI

Protein complex	*mr*	Subunits (Uniprot ID)
Cytoplasmic exosome complex	0.9	**P46948**,**P25359**,**P53859**,**P53256**,**Q12277**,**P38792**,**Q05636**,Q08162,**Q08285**,**P48240**
RSC complex	0.94	**P38781**,**Q12406**,**Q02206**,**Q06168**,**P25632**,**P53330**,P38210,**Q9URQ5**,**P53236**,**Q06639**,**P32832**,**Q05123**,**Q03124**,**P32597**,**Q07979**,**Q06488**,**P43609**
Mitochondrial ribosomal complex (small subunit)	0.81	**P53733**,P10663,**P47150**,**P21771**,**P38175**,**P10662**,**Q01163**,P02381,**P53305**,**Q02608**,**P36056**,**P12686**,**P32902**,**Q02950**,**P17558**,**Q03201**,**P27929**,**P47141**,P38796,**P28778**,P19955,**P40496**,**P33759**,**Q45TY3**,O75012,**Q03799**,**Q03246**,**Q03976**,**P38120**,**P53292**,**P42847**
TFIID complex	0.86	**Q03750**,P11747,P35189,**P46677**,**P38129**,**Q12030**,**P50105**,**Q12297**,**P23255**,**Q03761**,**P53040**,**Q05027**,**Q05021**,**Q04226**
mRNA cleavage and polyadenylation specificity factor complex	0.93	**P36104**,**P35728**,**P39927**,**P29468**,**Q06224**,**Q06632**,**Q12102**,P32598,**Q01329**,**P45976**,**Q08553**,**P42073**,**P42841**,**Q06102**,**P53538**
DASH complex	1.0	**P69852**,**P69851**,**P69850**,**P36131**,**P53168**,**Q12248**,**P36162**,**Q03954**,**P35734**,**P53267**
Anaphase-promoting complex	0.8	**P14724**,**Q12440**,**Q04601**,**P09798**,**Q08683**,**P53068**,**Q12157**,**P40577**,Q12379,**P38042**,**P53886**,**Q12107**,**P16522**,P26309,P53197
Nuclear exosome complex	1.0	**P46948**,**P25359**,**P53859**,**P38801**,**P53256**,**Q12277**,**P38792**,**Q05636**,**Q08162**,**Q08285**,**Q12149**,**P48240**
PBAF Complex	0.634	O96019,**Q68CP9**,**Q86U86**,**P51532**,**Q12824**,Q92922,**Q8TAQ2**,**Q969G3**,**Q96GM5**,Q92925,Q6STE5
Arp2/3 Complex	0.875	**P61160**,**P61158**,**O15143**,**O15144**,**O15145**,**P59998**,**O15511**

mr, matching rate between the identified protein complex and the known protein complex; Uniprot IDs of matched proteins are in bold font.

percolation, also identified DASH complex successfully, but P69850 and P69852 were excluded from the complex because of their lower connectivity, compared with other proteins in the complex. MCL identified all the proteins of DASH complex, but the identified structure involved a superfluous protein, Q12374 into the complex. Compared with other algorithms, EGCPI can identify complexes successfully may be due to the following reasons. First, it utilizes an evolutionary clustering approach to locating proteins sharing with a higher *degree of topological similarity* in a cluster. Then, the BFS method is used to form the protein complex not only based on topology but also the *degree of attribute homogeneity* between two connected proteins. For example,

there are 21 GO terms and 16 GO terms annotated to P36131 and P69852 as attributes values, respectively. 16 GO terms are shared by the two proteins so that *degree of attribute homogeneity* between them is 0.76, which is relatively high. Given this reason, P69852 is treated as a member of the identified protein complex by EGCPI. Taking into consideration attribute homogeneity when identifying protein complex makes EGCPI find more proteins with homogeneous attributes and improve its accuracy of protein complex identification.

In DIP-Hsapi, EGCPI successfully identified Arp2/3 Complex. The structure of Arp2/3 complex is shown in Fig. 16. As the figure shows, Arp2/3 complex is a typical star structure that O15144 connects all the other proteins in the protein complex. The identified protein complex successfully matched all the proteins in the known complex, but Q92747 was incorrectly incorporated into Arp2/3 complex. By investigating the GO terms annotated to Q92747 and O15144, we found that more than 60 percent of the attributes of Q92747 were also associated to O15144. As a result, EGCPI treated

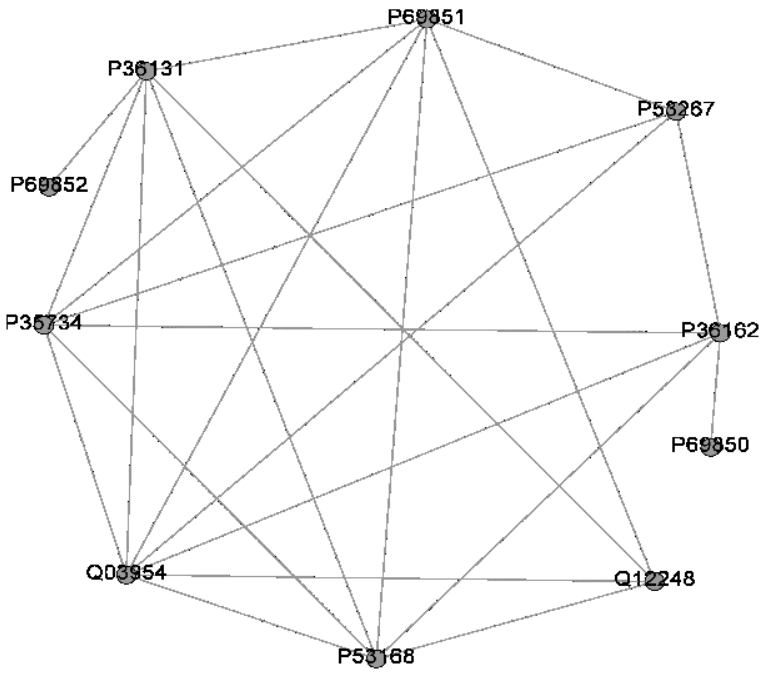

Fig. 15. The structure of DASH complex in CYC2008 database. The structure identified by EGCPI completely matches that of the DASH complex.

Q92747 as a member of the identified protein complex because Q92747 performs similar functions to those of O15144. Given the "incomplete" status of Arp2/3 complex and the similar functional attributes performed by Q92747 and O15144, there is a high possibility that Q92747 is confirmed as a member of Arp2/3 complex in future through laboratory-based experiments.

6.4. Summary

In this section, a novel approach to protein complex identification, EGCPI is proposed. EGCPI constructs a weighted PPI network Graph by assigning interactions with weights according to the *degree of the topological similarity* measure. Based on the evolutionary strategy that can form the optimal clusters with vertices that are densely connected in the PPI network graph, a breadth-first search method is then used to further partition each cluster and discover protein complexes with proteins sharing high *degree*

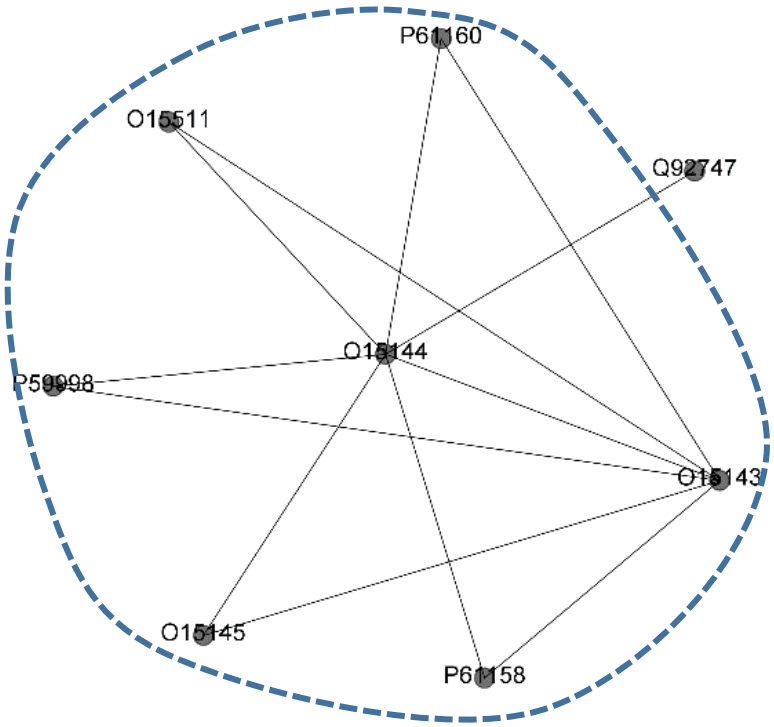

Fig. 16. The structure of Arp2/3 Protein complex in MIPS/CORUM database. The matched proteins identified by EGCPI are in the dashed circle.

of attribute homogeneity. The experimental performance proves that EGCPI using evolutionary graph clustering can obtain better results when identifying known protein complexes. In future, we will attempt to improve the efficiency of EGCPI, develop some evolutionary approaches which can discover overlapping protein complexes in the PPI network.

7. TBPCI-MINING CLUSTERS IN THE ATTRIBUTED GRAPH BY COMPUTING THE OPTIMAL DEGREE OF BOUNDEDNESS

7.1. Background

In this section, we present a novel approach, called TBPCI, to identify clusters based on the concept of a measure of boundedness. Such a measure is defined as an objective function of a Jaccard Index-based connectedness measure which takes into consideration how much two proteins within a network are connected to each other, and an association measure which takes into consideration how much two connecting proteins are associated based on their attributes. Based on the above two measures, the objective function is derived from capturing how strong the vertices can be considered as bounded together and the objective value is therefore referred as the aggregated degree of boundedness. To identify interesting sub-graphs, TBPCI computes the degree of boundedness between all possible pairwise vertices. Then, TBPCI uses a Breadth-First-Search method to determine whether a vertex pair should be incorporated into the same sub-graph. TBPCI has been used in the application of protein complex identification and tested with several real data sets and the experimental results show it is an effective approach for identifying clusters in the attributed graph.

To perform the task of protein complex identification in PPI network graphs, there have been several algorithms proposed. Among these methods, ones that consider both network topology and attribute information are not many. In [66], a method that identifies protein complexes based on finding clusters in which proteins perform similar functions is proposed. In [67], another algorithm is proposed to simultaneously consider PPI network data and gene expression data in the protein complex identification process. In [47], an algorithm called PCIA is proposed to identify protein complexes in PPI networks based on network topology and attribute information. It makes use first of a measure of attribute similarity followed by the use of the MCL algorithm to identify

densely connected clusters during the process. In [117], an algorithm called GMFTP, is proposed to identify protein complexes using the generative model by also considering both network topology and attribute information. The effectiveness of these algorithms shows that protein complexes can be more accurately identified when both topology and attribute information are considered.

Different from the state-of-the-art, TBPCI identifies interesting clusters as protein complexes in the PPI network graph based on information available about both attributes of the proteins and topology of their connections. Unlike traditional approaches, such as those based on graph partitioning, clique percolation, etc., the problem of protein identification is formulated as an optimization problem.

Given a PPI network graph, TBPCI performs its tasks by firstly computing the *Degree of Connectedness* (σ) which is defined to be between each pair of proteins in the PPI network. σ quantifies the extent that two proteins are similar based on the topology of the network. A higher value of σ can be interpreted as a higher degree of similarity between the local topology of the network that the proteins are in.

In addition to considering topology, TBPCI also measures how much the attribute values of two proteins are associated with each other. The measure, which we call, the *Degree of Attribute Association* (φ) is used to measure the strength of attribute correlation between each pair of proteins. The higher φ is, the stronger the attribute correlation between two proteins is.

After obtaining σ and φ for each pair of proteins in the PPI network, by representing the network in the form of a graph, TBPCI uses an iterative method to obtain an optimal weighted graph (**W**), each element of which measures how strong a pair of proteins can be bounded together. We name the entry in **W** as *Degree of Boundedness* (w) between

pairwise proteins. Using a Breadth-First-Search method in **W**, TBPCI can find graph clusters that have more tightly bound in a sense. These graph clusters are then considered to represent protein complexes. Given the experimental results obtained, TBPCI is shown to be a very promising method for mining meaningful clusters, when it is used in some of the real applications, e.g., protein complex identification in PPI network graphs.

7.2. TBPCI in details

7.2.1. Mathematical preliminaries

Given a PPI network containing n_V proteins and n_E interactions, it is represented as G = (V, E, Λ), where 1) V represents as the vertex set of the PPI network; 2) E represents as the edge set of the PPI network; 3) Λ is the set attribute values for proteins in G and it contains three subsets, Λ_p, Λ_f, Λ_c. They represent the sets of attributes of *biological processes*, *molecular functions* and *cellular components*, respectively.

It is also noted that the number of attributes from different domains that are associated with a vertex, say v_i, might be different. In general, $|\Lambda^i_p| \neq |\Lambda^i_f| \neq |\Lambda^i_c|$ for each vertex in G. Even sometimes the number of attributes from a particular domain might be equal to zero because the attribute information is missing or removed. As for different vertices, say v_i and v_j, the attributes from the same domain are always different, i.e. $|\Lambda^i_p| \neq |\Lambda^j_p|$, $|\Lambda^i_f| \neq |\Lambda^j_f|$ or $|\Lambda^i_c| \neq |\Lambda^j_c|$. TBPCI will perform the task of protein complex discovery using the given PPI network data G.

TBPCI needs to obtain the Degree of Connectedness and Degree of Attribute Association, σ and φ for pairwise proteins in the PPI network before identifying protein complexes. The information on σ and φ can be easily obtained by TBPCI, based on the data in G.

The *Degree of Connectedness* (σ) quantifies the extent that the common connected proteins are shared by two pairwise proteins, say v_i and v_j. It is defined as a Jaccard index similarity method:

$$\sigma_{ij} = \frac{\left|e_{i+} \cap e_{j+}\right| + e_{ij}}{\left|e_{i+} \cup e_{j+}\right| - e_{ij}} \tag{55}$$

where 1) e_{i+} is the set of edges that connect v_i and others; 2) the symbol |. | means the cardinality of a given set; 3) e_{ij} is equal to 1 if v_i and v_j are connected in G. For each element in σ, say σ_{ij}, it ranges between 0 and 1. A higher value of that means v_i and v_j share more common connected vertices in G. This indicates a higher similarity between v_i and v_j from the structure perspective.

Recently, some algorithms for protein complex identification, such as approaches proposed in [26], [27], [70], [71], are proposed based on identifying protein complexes whose proteins perform the homogeneous functions. Having deeply looked into different sets of real PPI network data, we found that those proteins with functional homogeneity belong to the same protein complexes was not always the truth. Let the molecular function positive regulation of transcription from RNA polymerase II promoter (GO:0045944) be an example. In some real PPI network data, such as the one described in [105], the number of proteins that perform the above molecular function is 250. Apparently, it is a relatively high number. As a result of using functional similarity as the measure of identifying protein complexes, these 250 proteins might be located in the same complex. Therefore, a probable outcome of such a grouping method is a lower rate of identification. Moreover, there is another observed fact that some known protein complexes are constituted of proteins which perform very different functions. For example, one protein with ID P50874 of nuclear origin recognition complex (GO: 0005664) performs very different functions, compared with other proteins in that

protein complex. As a result, those algorithms based on functional homogeneity probably cannot identify such protein complexes whose proteins perform different functions. Given such observed facts, to measure the correlation between pairwise proteins from the property of attribute information, TBPCI computes the *degree of attribute association* (φ) using (2), (6), and (7).

After obtaining all the values of φ, TBPCI uses values of θ when the corresponded σ is larger than 0. TBPCI makes use of two matrices, **S** and **A** to store the obtained values of σ and φ between pairwise vertices in G. TBPCI considers to find optimal weights for those pairwise vertices with a larger-than-zero σ when identifying protein complexes in a PPI network. The found weight measures how strong two pairwise proteins can be bounded together.

7.2.2. Finding optimal weights between pairwise vertices

7.2.2.1. *Objective function and updating method*

Given the obtained information on **S** and **A** and the PPI network data G, TBPCI attempts to seek optimal weights that may quantify the strength that each pair of proteins can be considered as bounded together. Such weight is named as *Degree of Boundedness* (w). To complete the task, TBPCI formulates the following optimization problem:

$$\begin{aligned} &\textit{Maximize} \\ &O(\mathbf{W},\mathbf{F}) = \mathbf{tr}\left(\mathbf{W}_{\mathbf{S}}^{\mathrm{T}}\mathbf{W}\right) + \alpha\mathbf{tr}\left(\mathbf{W}_{\mathbf{A}}^{\mathrm{T}}\mathbf{W}\right) - \left|\mathbf{W}\right|_F^2 - \left|\mathbf{F}\right|_F^2 \\ &\mathbf{W}_{\mathbf{S}} = \mathbf{W} \circ \mathbf{S},\ \mathbf{W}_{\mathbf{A}} = \mathbf{F} \circ \mathbf{A} \\ &\textit{subject to}\ \ 0 \le \mathbf{W} \le 1, 0 \le \mathbf{F} \le 1 \end{aligned} \tag{56}$$

where 1) **W** is the matrix in which the variables represent the Degree of Boundedness between pairwise vertices, 2) **F** is the matrix in which the variables represent the weight between pairwise vertices when the attribute association is considered only, 3) the symbol "$\circ$" means the Hadamard product of two matrices, 4) α is a parameter that is

used to constrain the effect of attribute association within the optimization process. The objective value of (56) aggregates the Degree of Boundedness that each pair of vertices in G with a larger-than-zero σ. Apparently, those pairwise vertices with relatively higher values of σ and φ should be assigned larger ws. However, obtaining appropriate Degrees of Boundedness between all possible pairwise vertices cannot be achieved immediately. In order to obtain optimal Degrees of Boundedness, TBPCI uses an iterative method that constantly updates the values in **W** and **F** till the objective function achieves convergence. The updating methods for **W** and **F** used by TBPCI are

$$\begin{aligned}
&keeping\ \ \mathbf{F}: \\
&w_{ij} \leftarrow \begin{cases} [\mathbf{W}+\eta\Delta\mathbf{W}]_{ij} & if\ 0 \le [\mathbf{W}+\eta\Delta\mathbf{W}]_{ij} \le 1 \\ \Omega([\mathbf{W}+\eta\Delta\mathbf{W}]_{ij}) & otherwise \end{cases} \quad (a) \\
&\Delta\mathbf{W} = 2\mathbf{W}_{\mathbf{S}} + \alpha\mathbf{W}_{\mathbf{A}} - 2\mathbf{W} \\
&keeping\ \ \mathbf{W} \\
&f_{ij} \leftarrow \begin{cases} [\mathbf{F}+\delta\Delta\mathbf{W}_{\mathbf{A}}]_{ij} & if\ 0 \le [\mathbf{F}+\delta\Delta\mathbf{F}]_{ij} \le 1 \\ \Omega([\mathbf{F}+\delta\Delta\mathbf{F}]_{ij}) & otherwise \end{cases} \quad (b) \\
&\Delta\mathbf{F} = \alpha(\mathbf{W}\circ\mathbf{A}) - 2\mathbf{F}
\end{aligned} \tag{57}$$

where η and δ are the step lengths for each iteration of updating for **W** and **F**, respectively. By alternatively updating **W** and **F** following the rules (57a) and (57b), the objective function in (56) will be lead to the convergence of the local optima.

7.2.2.2. *Convergence analysis*

Whether the proposed objective function is convergent is essential for TBPCI to identify protein complexes in the PPI network. Here we will give a detailed proof for the convergence of the objective function (56), using the updating method in (57). First, the proposed objective function is proved to be convergent without constraint. And then, we will proof that the objective function is also convergent when the box constraints, i.e. $0 \le \mathbf{W} \le 1$ and $0 \le \mathbf{F} \le 1$ are used by identifying the relationship of convergence between the above two types of constraints.

For the proof of convergence of (56) by updating **W** following the rule in (57a), it is equivalent to show that, keeping **F** unchanged, $O(\mathbf{W}^{t+1}, \mathbf{F}) \geq O(\mathbf{W}^{t}, \mathbf{F})$ after updating **W** following (57a) in each iteration. Since the updating rule is elementwise, it is sufficient to show for any element w_{ij}, $O(w_{ij})$ is non-decreasing, under the updating rule in (57). Therefore, we have

$$\begin{aligned} O(w_{ij}^{t+1}) &= O(w_{ij}^{t} + \eta \Delta w_{ij}^{t}) \\ \Delta w_{ij}^{t} &= \left[2\mathbf{W_S} + \alpha \mathbf{W_A} - 2\mathbf{W}\right]_{ij} \end{aligned} \tag{58}$$

Here we assume that the step length η is a small positive scalar which is smaller than 1 and near to zero. Hence, we can conclude that w_{ij}^{t+1} is near to w_{ij}^{t}, after the updating. To investigate the local information near to w_{ij}^{t}, we use the Taylor series expansion to rewrite O with respect to w_{ij} as

$$O(w_{ij}^{t+1}, w_{ij}^{t}) = O(w_{ij}^{t}) + \frac{\partial O}{\partial w_{ij}^{t}}(w_{ij}^{t+1} - w_{ij}^{t}) + \frac{1}{2}\frac{\partial^2 O}{\partial \left(w_{ij}^{t}\right)^2}(w_{ij}^{t+1} - w_{ij}^{t})^2 \tag{59}$$

Since w_{ij}^{t+1} is near to w_{ij}^{t}, the value of (59) can be seen as an approximant of (56) after updating w_{ij} from iteration t to t+1. If the sum of the latter two components in (59) is non-negative, we can verify that O is non-decreasing when updating w_{ij} according to (57a). Because w_{ij} is any element in **W**, it also means O is finally convergent with respect to **W**. It is noted that (59) can also be written as

$$O(w_{ij}^{t+1}, w_{ij}^{t}) = O(w_{ij}^{t}) + \frac{\partial O}{\partial w_{ij}^{t}}\eta \Delta w_{ij}^{t} + \frac{1}{2}\frac{\partial^2 O}{\partial \left(w_{ij}^{t}\right)^2}\eta^2 \left(\Delta w_{ij}^{t}\right)^2 \tag{60}$$

By replacing Δw_{ij}, first-order and second-order partial derivative of O, the last two components can be written as

$$\frac{\partial O}{\partial w_{ij}^{t}}\eta\Delta w_{ij}^{t}+\frac{1}{2}\frac{\partial^{2}O}{\partial\left(w_{ij}^{t}\right)^{2}}\eta^{2}\left(\Delta w_{ij}^{t}\right)^{2}=\left(\frac{\partial O}{\partial w_{ij}^{t}}\right)^{2}\left(\frac{1}{2}\eta^{2}\frac{\partial^{2}O}{\partial\left(w_{ij}^{t}\right)^{2}}+\eta\right)$$
$$=\left[2\mathbf{W}_{\mathbf{S}}+\alpha\mathbf{W}_{\mathbf{A}}-2\mathbf{W}\right]_{ij}^{2}\left[\left[\mathbf{S}-\mathbf{1}\right]_{ij}\eta^{2}+\eta\right] \tag{61}$$

Hence the sum of last two components in (59) is equal to the product of two scalar shown in (61). Obviously, the first scalar is non-negative. According to our assumption that the step length η is a small positive scalar which is smaller than 1 and near to zero, and $s_{ij}\leq 1$, we conclude η^2 is smaller than η and $[\mathbf{S}\text{-}\mathbf{1}]_{ij}\eta^2+\eta>0$. Thus, we have

$$O(w_{ij}^{t+1},w_{ij}^{t})-O(w_{ij}^{t})=\frac{\partial O}{\partial w_{ij}^{t}}(w_{ij}^{t+1}-w_{ij}^{t})+\frac{1}{2}\frac{\partial^{2}O}{\partial\left(w_{ij}^{t}\right)^{2}}(w_{ij}^{t+1}-w_{ij}^{t})^{2}\geq 0 \tag{62}$$

This means $O(w_{ij}^{t+1})$ is non-decreasing and $O(w_{ij}^{t+1})= O(w_{ij}^{t})$ only when O converges to the local optima. Next, we will show the proposed objective function can also achieve its local optima when the box-constraint $0\leq\mathbf{W}\leq 1$ is active. Based on the KKT condition, when (56) is convergent with respect of $\mathbf{W}$, we have the following

$$\begin{cases} w_{ij}=1 & \dfrac{\partial O}{\partial w_{ij}^{t}(w_{ij}^{t}=1)}\geq 0 \\ 0<w_{ij}<1 & \dfrac{\partial O}{\partial w_{ij}^{t}}=0 \\ w_{ij}=0 & \dfrac{\partial O}{\partial w_{ij}^{t}(w_{ij}^{t}=0)}\leq 0 \end{cases} \tag{63}$$

It is apparent that (56) is convergent when w_{ij} satisfies the second case in (63) since its first order partial derivative at w_{ij} is zero and we have proved that it is convergent when $\mathbf{W}$ is updated following (57a). Hence, it is essential to investigate whether (56) is convergent when w_{ij} achieves the upper or lower boundary. Assume w_{ij} is any element in $\mathbf{W}$ and it equals to 1, the first-order partial derivative of (56) about w_{ij} at 1 is larger than zero. This indicates (56) achieves its local optima when $w_{ij}>1$. We also assume that

there is another point w'_{ij}, that is within the constrained area, near to w_{ij} and leads (56) to a larger objective value. Based on the above assumptions, we have

$$\begin{aligned} O(w'_{ij}, w_{ij}) - O(w_{ij}) &= \frac{\partial O}{\partial w_{ij}}(w'_{ij} - w_{ij}) + \frac{1}{2}\frac{\partial^2 O}{\partial (w_{ij})^2}(w'_{ij} - w_{ij})^2 \\ &= (w'_{ij} - 1)\left[(\mathbf{S}-\mathbf{1})_{ij}\left(1 + w'_{ij}\right) + \alpha \mathbf{W}_{\mathbf{A}ij}\right] > 0 \end{aligned} \tag{64}$$

It is noted that $(w'_{ij}-1) < 0$. Thus, the above inequality holds only when the second component is smaller than zero. But, by comparing the second component of the inequality with the first order partial derivative of (56) about w_{ij}, we have

$$(\mathbf{S}-\mathbf{1})_{ij}\left(1 + w'_{ij}\right) + \alpha \mathbf{W}_{\mathbf{A}ij} > 2(\mathbf{S}-\mathbf{1})_{ij} + \alpha \mathbf{W}_{\mathbf{A}ij} \tag{65}$$

As we have assumed that the first-order partial derivative of (56) about w_{ij} at 1 is larger than zero, the result obtained in (64) contradicts to the assumption. Therefore, for the first case in (63), (56) converges with respect to w_{ij}=1. As w_{ij} is any element in **W**, (56) is convergent with respect to **W**.

For the case that w_{ij} at the lower boundary, we have the following assumptions before proving its convergence: w_{ij} is any element in **W** and it equals to 0, the first-order partial derivative of (56) about w_{ij} at 0 is smaller than zero. This indicates that (56) achieves its local optima at somewhere w_{ij} <0. We also assume that there is another point w'_{ij}, that is within the constrained area, near to w_{ij} and leads (56) to a larger objective value. Based on the above assumptions, we have the following

$$O(w'_{ij}, w_{ij}) - O(w_{ij}) = \frac{\partial O}{\partial w_{ij}} w'_{ij} + \frac{1}{2}\frac{\partial^2 O}{\partial (w_{ij})^2} w'^{\,2}_{ij} > 0 \tag{66}$$

As w'_{ij}>0, the second order partial derivative of (56) about w_{ij}=0 is non-positive, (66) holds only when the first-order partial derivative of (56) about w_{ij} at 0 is larger than

zero. But this contradicts to the assumption that the first-order partial derivative of (56) about w_{ij} at 0 is smaller than zero. Therefore, for the third case in (63), (56) converges with respect to w_{ij}=0. As w_{ij} is any element in **W**, (56) is convergent with respect to **W**.

In a word, following the updating rule for **W** in (57a), the objective function in (56) will converge to its local optima with respect to **W**.

As the proof of convergence of (56) following the updating rule for **F** is very similar to that of **W**, we don't present the proof in detail. The updating rule for **F** in (57) ensures the objective function in (56) is non-decreasing and it can achieve its local optima when the box constraint $0 \leq \mathbf{F} \leq 1$ is active. Therefore, by keeping one vector of variables unchanged and updating the other, we have

$$O(\mathbf{W}^0, \mathbf{F}^0) \leq O(\mathbf{W}^1, \mathbf{F}^0) \leq O(\mathbf{W}^1, \mathbf{F}^1) \leq \ldots \leq O(\mathbf{W}^*, \mathbf{F}^*) \tag{67}$$

Algorithm 5
Seeking optimal Degree of Boundedness

Input: **S**, **A**, α, η, δ, τ, *MaxIteration*
Output: **W, F**

```
Randomly initialize W and F;
Compute objective value
t←1;
do
{ oW←W;
   update W←W+η∆W according to rule (57a);
   update F←F+δ∆F according to rule (57b);
   compute objective value;
   t←t+1;
}while((W-oW)^T(W-oW) >θ and t< MaxIteration);

return W, F;
```

Fig. 17. Pseudo codes of the optimization process

This non-decreasing updating will stop till the objective function (56) converges to the local optima $O(\mathbf{W}^*, \mathbf{F}^*)$.

7.2.2.3. *Stopping criterion for the optimization process*

As the rate of updating becomes lower as **W** and **F** approach to $\mathbf{W}^*$ and $\mathbf{F}^*$, we use the following criterion to determine whether the optimization process stops at after a certain number of iterations:

$$(\mathbf{W}^t - \mathbf{W}^{t-1})^{\mathrm{T}}(\mathbf{W}^t - \mathbf{W}^{t-1}) \leq \tau \tag{68}$$

where τ is a positive number to tighten or loose the minimum rate of updating **W** that satisfies the stopping condition.

Algorithm 6
Protein complex identification

```
Input:    G, W
Output:   Protein complex set
reconstruct W according to G;
for each vertex v_i in W{
   create linked list visiting and visited;
   find hw_i;
    if(hw_i>0){
      add v_i to visiting;}
      while(visiting ≠ Φ){
          v_j←head of visiting;
          delete v_j from visiting;
          add v_j to visited;
          search v_k: neighbors of v_j;
          if(W_jk ≥ hw_i){
                add v_k to visiting;}}
        if(|visited|>0){
          create protein complex PC_i;
          if(Max_OSPCi<Max_OS){
   add PC_i to Protein complex set;}}}
 return Protein complex set;
```

Fig. 18. Protein complex identification

7.2.2.4. *Summary of the optimization process*

Based on the above description on finding the optimal Degree of Boundedness between each pair of proteins in the PPI network, the approach can be summarized as the pseudo codes shown in Fig. 17.

7.2.3. Identifying sub-graphs in the weighted graph

Having obtained the optimal **W**, TBPCI sets all those variables in **W** to 0 if the corresponding vertices in G are not connected. After that, **W** can be considered to be a weighted graph with each element (w_{ij}) in it representing the strength that vertex v_i and v_j are bounded together. w_{ij} is larger than zero when there is an edge between v_i and v_j in G and they have a certain degree of boundedness. A higher value of w_{ij} means a higher boundedness that v_i and v_j can be grouped together. TBPCI uses **W** to complete the task of identifying protein complexes in the PPI network.

Given **W**, TBPCI performs a further search of graph clusters as protein complexes. To perform the task of protein complex identification, TBPCI uses a Breadth-First-Search (BFS) method to form protein complexes by selecting each protein in the PPI network as a seed. First, it selects an edge with the highest weight hw_i, that connects the seed in **W**, and incorporates both of the two connected vertices v_i and v_j into a set for forming a protein complex; second, based on the weight of the selected edge, TBPCI searches all the neighboring vertices and incorporates those which satisfy the minimum threshold of w. In TBPCI, this threshold is defined as:

$$PC(seed : v_k) = \begin{cases} PC \cup v_m \ if \ w_{km} \geq \lambda \times hw \\ PC \cup \Phi \ otherwise \end{cases} \tag{69}$$

where v_k stands for any vertex in the set and v_m is any vertex connecting to v_k. In other words, λ is used to tighten or loose the minimum w. Only vertices sharing connections with ws that are not lower than $\lambda \times hw_i$ can be incorporated into the complex set so that

all the proteins in the complex are tightly bounded to each other. The searching in the second step will be terminated till there is no new vertex added. When the above search is finished, TBPCI forms a protein complex constituted by the connected vertices selected in the searching phase. To reduce the redundancy between the identified protein complexes, TBPCI uses another measure, *Maximum Overlapping Score* to finally determine whether the identified protein complex should be incorporated into the set of identified protein complexes. And the *Maximum Overlapping Score* is defined as:

$$Max_{OS} = \max \frac{|PC_I \cap PC_S|}{|PC_I \cup PC_S|} \tag{70}$$

where PC_I and PC_S stand for the identified protein complex and any protein complex in the complex set, respectively. When Max_{OS} is larger than some threshold, TBPCI will not incorporate the identified protein complex into the identified complex set. A lower threshold of Max_{OS} used by TBPCI means there are fewer same proteins formulating each protein complex in the identified set. TBPCI will stop forming protein complexes till it traverses all vertices in **W**. In order to explain this BFS method in detail, we give its pseudo codes in Fig. 18. It also should be noted that any method which can effectively extract connected components in the weighted graph can be used to discover protein complexes in the weighted graph constructed by TBPCI.

7.3. Experiment and analysis

For performance testing, TBPCI has been tested with five sets of real world PPI network data: Collins [23], Gavin [37], Krogan [56], DIP-Scere [105], and DIP-Hsapi [105]. The detail information on these datasets have been summarized in Section 6.3.

For the experiments, the attributes information on the proteins were obtained from GO database [24]. As what has been mentioned, all GO terms in *cellular components* which

TABLE 15 PARAMETER SETTINGS OF DIFFERENT APPROACHES

Approach	Parameter	Approach	Parameter
TBPCI	Experiment Trials	PCIA	inflation=1.8, μ=0.7 (default setting)
MCL	inflation = 1.8 (default setting)	GMFTP	K=1000 (default setting)
RNSC	N/A	CFinder	K=3
MCODE	VWP=0.2 (default setting)	COACH	Experiment Trials
CMC	Experiment Trials		

N/A: There is no parameters needed to be set.

may infer a particular protein belongs to some complex (es) have been excluded.

For performance evaluation using the four sets of data related to *yeast Saccharomyces cerevisiae*, we compared the protein complexes identified by different algorithms with the known protein complexes as contained in the CYC2008 [83] and the MIP/CYGD [42] [75] databases. The known protein complexes in these two databases were both collected in March 2013. Altogether, there are 408 known protein complexes in CYC2008 and 255 known protein complexes in MIP/CYGD database. After merging the data in the two databases, a total of 296 different known protein complexes containing more than 2 proteins were available for our use for performance analysis.

For the DIP-Hsapi data set, the protein complexes identified in it by each algorithm were compared with known protein complexes in the MIPS/CORUM [88] database which contain a total of 1466 different protein complexes.

7.3.1. Experimental set-up and evaluation measurement

For performance evaluation, TBPCI were compared with a number of different algorithms, including PCIA, MCL, GMFTP, RNSC, MCODE, CFinder, CMC and COACH. When testing these algorithms, we either used the default parameter settings recommended by the authors or tried as many different settings as we can to obtain the best performance with different algorithms. For the case of TBPCI, we set α= 0.9, η=δ=0.05 τ=1e-6 and *MaxIter* = 100. For λ and Max_{OS}, they are set within the range

TABLE 16 EXPERIMENTAL RESULTS OF *F-MEASURE* AND *MMR* OBTAINED FROM BIOGRID DATASETS

Data Set	*Approach*	*#*	*Coverage*	*Precision*	*Recall*	*f-measure*	*MMR*
Collins	TBPCI	392	1194	0.599	0.642	**0.62**	**0.32**
	PCIA	416	1607	0.368	0.591	0.453	0.303
	GMFTP	203	1088	0.596	0.473	0.527	0.283
	MCL	282	1620	0.401	0.493	0.442	0.259
	MCODE	111	862	0.775	0.357	0.489	0.208
	RNSC	353	1480	0.365	0.544	0.437	0.309
	CFinder	114	1160	0.702	0.319	0.438	0.196
	CMC	200	1067	0.545	0.443	0.489	0.265
	COACH	245	1114	0.51	0.481	0.495	0.176
Gavin	TBPCI	404	1191	0.619	0.607	**0.613**	**0.247**
	PCIA	378	1417	0.315	0.438	0.366	0.216
	GMFTP	172	843	0.643	0.421	0.509	0.228
	MCL	177	1430	0.395	0.278	0.336	0.145
	MCODE	66	602	0.682	0.167	0.268	0.099
	RNSC	307	1258	0.322	0.402	0.358	0.221
	CFinder	98	1124	0.561	0.2	0.295	0.115
	CMC	391	946	0.263	0.392	0.315	0.203
	COACH	324	1052	0.43	0.46	0.445	0.219
Krogan	TBPCI	863	2083	0.453	0.762	**0.547**	**0.342**
	PCIA	1022	2633	0.158	0.583	0.248	0.332
	GMFTP	299	1376	0.413	0.492	0.449	0.282
	MCL	514	2674	0.2	0.422	0.272	0.224
	MCODE	75	552	0.693	0.2	0.31	0.109
	RNSC	751	2142	0.169	0.5	0.253	0.298
	CFinder	115	1140	0.496	0.213	0.298	0.139
	CMC	869	1100	0.236	0.592	0.337	0.287
	COACH	349	1056	0.494	0.536	0.514	0.275

#: The number of protein complexes identified. Coverage: The number of distinct proteins in the identified protein complexes.

from 0.1 to 1 with a 0.1 increment while the performance was being tuned. The detailed settings of parameters for all different algorithms are listed in Table 15.

For performance comparison, we mainly used two different measures, *f-measure*, and *Maximum Matching Rate* (*MMR*), which have been defined in Section 6.3. Based on their definitions, *f-measure* and *MMR* can be seen to complement each other.

Besides evaluating the experimental results by *f-measure* and *MMR*, we also used GO::TermFinder [15] to perform functional enrichment analysis on the protein complexes identified by TBPCI. This analysis helps to evaluate the biological significance of the protein complex identified by TBPCI. Some of the results of the

TABLE 17 EXPERIMENTAL RESULTS OF *F-MEASURE* AND *MMR* OBTAINED FROM DIP DATASETS

Data Set	*Approach*	#	*Coverage*	*Precision*	*Recall*	*f-measure*	*MMR*
Dip-Scere	TBPCI	1405	3307	0.348	0.836	**0.491**	0.349
	PCIA	1497	4445	0.117	0.636	0.198	0.328
	GMFTP	514	2509	0.286	0.605	0.388	0.305
	MCL	691	4579	0.124	0.327	0.18	0.184
	MCODE	60	756	0.383	0.082	0.135	0.046
	RNSC	1376	3772	0.094	0.522	0.16	0.284
	CFinder	192	2143	0.302	0.205	0.244	0.129
	CMC	1389	1763	0.184	0.736	0.295	**0.353**
	COACH	854	1952	0.275	0.697	0.395	0.32
Dip-Hsapi	TBPCI	754	1822	0.336	0.184	**0.237**	**0.067**
	PCIA	706	2179	0.189	0.108	0.138	0.063
	GMFTP	187	806	0.31	0.044	0.077	0.024
	MCL	549	2434	0.171	0.074	0.103	0.044
	MCODE	61	273	0.459	0.019	0.038	0.008
	RNSC	730	1847	0.148	0.086	0.109	0.059
	CFinder	111	515	0.505	0.042	0.077	0.019
	CMC	149	403	0.537	0.061	0.109	0.027
	COACH	151	492	0.629	0.073	0.131	0.029

analysis are presented in the following sections.

7.3.2. Performance analysis

The results of the different experiments performed using the different data sets and algorithms are presented in Table 16 and 17 in terms of the values of the *f-measure* and *MMR*. As shown in the Table, according to the *f-measure*, TBPCI performed the best in all the five data sets. When evaluated with *MMR*, TBPCI also obtains a consistently good performance. In terms of the number of identified protein complexes and coverage, it is noted that TBPCI records a relatively higher coverage while maintains a smaller number of identified protein complexes when compared with the other algorithms.

7.3.3. Convergence of optimization process

It is shown earlier that the objective function (56), that TBPCI uses may achieve local optima by iteratively updating **W** and **F** based on (57a) and (57b). In performing the experiments, the variations of the objective values were tracked. In all experiments using different data sets, it was noted that the objective values exhibit similar changing trends. In Fig. 19, the curve showing the changes in objective value which obtained

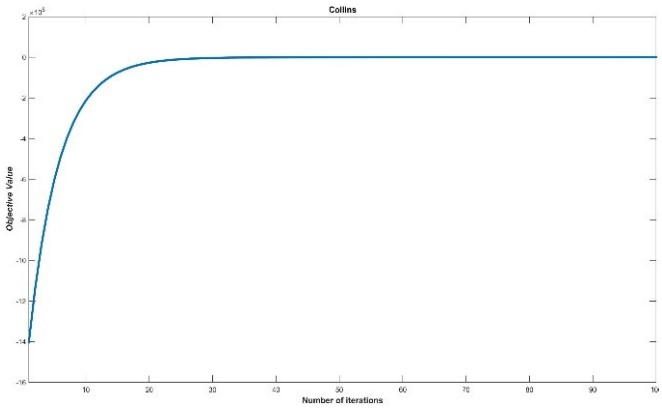

Fig. 19. Curve on the variations of objective values

from the data set Collins is displayed. As seen in the figure, TBPCI could approach a convergent objective value within a number of iterations. This also means that the updating rate of **W** becomes less evident as the number of iterations increases. When the updating rate is less than the predefined threshold or TBPCI achieves the maximum number of iterations, the **W** obtained can be considered as optimal Degrees of Boundedness between pairs of proteins in a PPI network.

7.3.4. Function enrichment analysis between TCPCI and PCIA

Besides evaluating TBPCI using the *f-measure* and *MMR*, we have also tried to find out whether there was something biologically significant in the identified protein complexes. To do so, we used GO::TermFinder [15] to make a functional enrichment analysis. Provided by SGD [25], GO::TermFinder is a web-based service that can be used for searching significant shared GO terms in the proteins of an identified protein complex. To perform more detailed analysis, we used different thresholds of *p-value*s when analyzing the identified protein complexes. In other words, those GO terms whose *p-values* are equal to or lower than the threshold may be identified as significant ones.

To compare the performance between the approaches that make use of attributes information when they identify protein complexes in the PPI network, we also used

TABLE 18 RESULTS OF FUNCTIONAL ENRICHMENT ANALYSIS OBTAINED BY TBPCI AND PCIA

Data Set	Approach	≤1.0E-10	≤1.0E-5	≤1.0E-2
Collins	TBPCI	197(50.26%)	330(84.18%)	369(94.13%)
	PCIA	107(25.72%)	266(63.94%)	334(80.29%)
Gavin	TBPCI	288(71.29%)	379(93.81%)	399(98.76%)
	PCIA	98(25.92%)	198(52.38%)	268(70.89%)
Krogan	TBPCI	347(40.21%)	679(78.68%)	789(91.43%)
	PCIA	118(11.55%)	336(32.88%)	604(59.09%)
DIP-Scere	TBPCI	651(46.33%)	1126(86.11%)	1318(93.8%)
	PCIA	158(10.55%)	424(28.32%)	848(56.65%)
DIP-Hsapi	TBPCI	325(43.1%)	663(87.93%)	743(98.54%)
	PCIA	135(19.12%)	484(68.56%)	610(86.4%)

X (Y%): the number of identified protein complexes that share significant GO terms under a particular threshold and the proportion of protein complexes sharing significant GO terms in the total protein complexes identified by each algorithm.

GO::TermFinder to analyze the protein complexes identified by PCIA, which has been proved to be a very effective approach to protein complex identification. It should be noted that not all the proteins in these protein complexes share significant GO terms that are known and can be found in databases such as MIPS/CYGD, CYC2008, and MIPS/CORUM. In such cases, they can be considered as candidates of real protein complexes due to their statistical significance revealed by the function enrichment analysis. The results of functional enrichment analysis on the protein complexes that are identified by TBPCI and PCIA are summarized in Table 18.

As the table shows, the protein complexes identified by TBPCI obtain a much better performance on the functional enrichment analysis. This means that there are many more protein complexes identified by TBPCI that have higher possibility to be real protein complexes that are yet to be confirmed, although TBPCI doesn't perform better with the *f-measure* or *MMR*, just like *MMR* in the case of DIP-Scere.

Based on the results of functional enrichment analysis, we can conclude that TBPCI is a very promising approach for protein complex identification.

7.3.5. Examples of protein complexes identified by TBPCI

We select several protein complexes identified by TBPCI to determine how the consideration of both topology and attribute information may allow interesting protein complexes to be identified.

One example of protein complex that is identified by TBPCI is Kornberg's mediator (SRB) complex (MIPS ID 510.40.20). This protein complex was identified by TBPCI in the data set Krogan. Its structure is shown in Fig. 20. 16 proteins out of 20 that are the same as known protein complexes were successfully identified. For other approaches, which also identified protein complexes similar to the SRB complex, such as CMC and COACH, missed some proteins have fewer interactions, such as Q99278, with other proteins. MCODE was not able to successfully identify this protein complex.

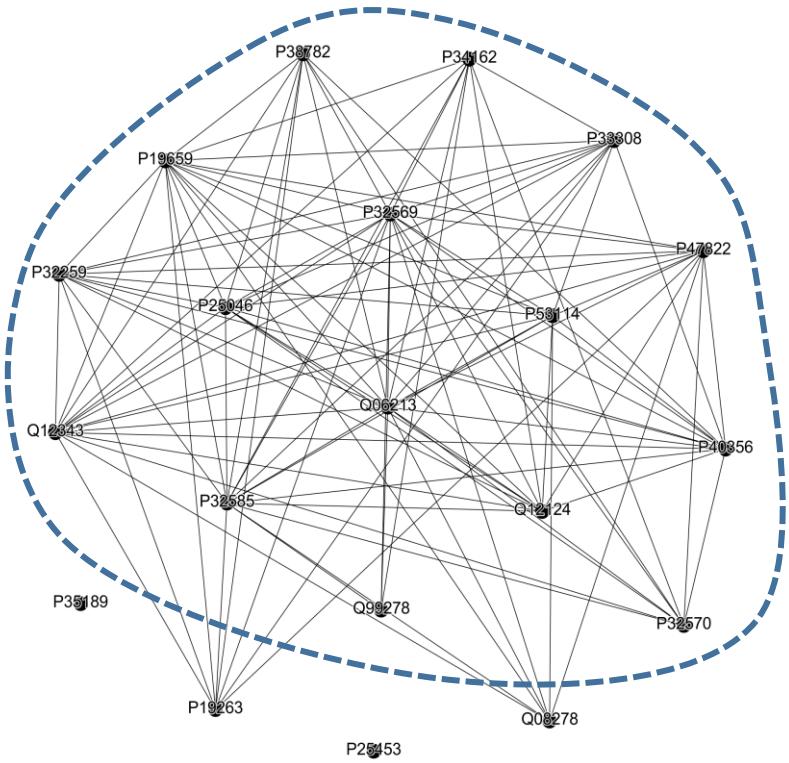

Fig. 20. The structure of Kornberg's mediator (SRB) complex (MIPS ID: 510.40.20) in the MIPS/CYGD database. The proteins successfully identified by TBPCI are circled in the dashed line.

Although proteins like Q99278 may share fewer interactions with other proteins in the SRB complex, we find that it has a relatively high Degree of Attribute Association with other proteins. Hence, the Degree of Boundedness assigned by TBPCI is appropriate for Q99278 to be identified as a member of the SRB complex.

It should be noted that two proteins, Q08278 and P19263 were not correctly identified by TBPCI as they share relatively lower Degree of Connectedness and Attribute Association with other identified proteins. Hence, they are assigned with lower degrees of Boundedness by TBPCI and may therefore not satisfy the threshold of λ to be incorporated into the complex. As for the other two proteins, P25453 and P35189, they are not connected to any protein in the SRB complex and are therefore not identified by TBPCI.

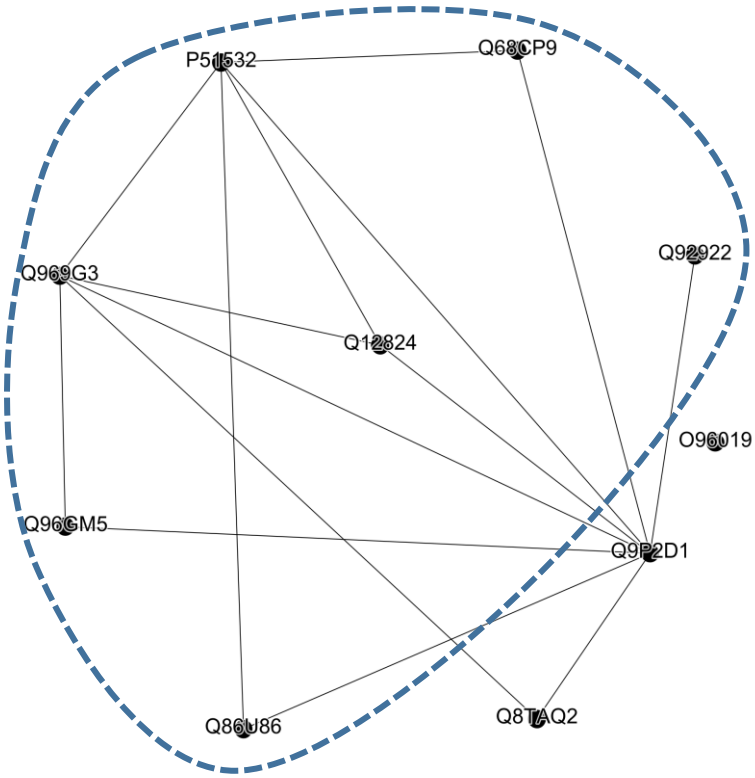

Fig. 21. The protein structure of PBAF complex (CORUM ID: 1238) in MIPS/CORUM database. Proteins successfully identified by TBPCI are in the dashed line.

When considering both topology and attribute information of a PPI network, TBPCI may find an appropriate Degree of Boundedness that can be used to quantify the relationship and identify those proteins with fewer interactions with other proteins such as Q99278 as a member of some protein complex.

Another example of an identified protein complex is discovered in DIP-Hsapi. In that data set, TBPCI successfully identified the PBAF complex (CORUM ID: 1238). The structure of the PBAF complex is shown in Fig. 21. Seven out of nine proteins in the PBAF complex were detected by TBPCI. What is interesting about the structure of the PBAF complex is O96019. Based on the interactions obtained from the dataset DIP-Hsapi, it should be noted that it is disconnected from all other proteins in PBAF. Given such a reason, TBPCI did not treat it as a member of the PBAF complex.

The protein Q9P2D1 is also worthy of notice as TBPCI identified it as a member of the PBAF complex. In fact, it shares relatively high Degrees of Connectedness and Attribute Association with other proteins in the PBAF complex and it is the only protein that interacts with Q92922 which is a member of the PBAF complex but shares no interactions with other proteins. Given the structure and Attribute between Q9P2D1 and other proteins in PBAF complex, we can conclude that Q9P2D1 is possible to be verified as a member of the complex in future laboratory-based experiments.

As for the other missing protein, Q8TAQ2, we find that it is very different from other proteins in the PBAF complex from both the structural and attributional perspectives. This indicates that Q8TAQ2 has fewer interactions with other proteins and its attributes are not well-related to those of the other proteins. Hence, the Degrees of Boundedness computed by TBPCI are smaller than the others in the PBAF complex and Q8TAQ2 was excluded from the complex by TBPCI.

Based on the identified protein complex and the known PBAF complex, it can be said that TBPCI might identify the two missing proteins if there is more evidence showing that Q8TAQ2 and O96019 are related to other proteins in PBAF, either from the topological or the attributional perspectives.

7.4. Summary

In this section, a novel approach to mining sub-graphs in the attributed graph, TBPCI is proposed. TBPCI considers seeking appropriate Degrees of Boundedness between pairwise vertices in attributed graph by taking both topology and attribute information into the consideration. Based on the optimization strategy that can find the optimal Degrees of Boundedness that may quantify the strength that pairwise vertices can be bounded together, a Breadth-First-Search method is then used to search sub-graphs in the weighted graph that is constructed based on the found Degrees of Boundedness. TBPCI has been used to identify sub-graphs in the PPI network graphs. The experimental performance proves that TBPCI using such an optimization strategy can obtain better results when identifying known protein complexes. In future, we will attempt to improve the efficiency of TBPCI, develop some approaches discovering overlapping protein complexes in the PPI network.

8. CONCLUSION

8.1. Summary

In this thesis, we propose to use *Attribute Graph* to model the relational data and to perform the task of discovering clusters, or communities which are interesting sub-graphs in which vertices are cohesively inter-related. Aiming at addressing the challenges existing in the state-of-the-art of the graph clustering algorithms, we propose four different algorithms, which utilize different techniques but take into consideration attribute and topology, to detect meaningful clusters in the attributed graph. To identify an optimal cluster arrangement, we propose to use MISAGA, which may formulate the task of cluster detection as a constrained optimization problem and solve it by using an iterative updating method, to find clusters in the attributed graph. As some clusters, might be overlapping, e.g., social communities in social graphs, and functional modules in biological graphs, we propose FSPGA, which formulates the task of cluster detection as a fuzzy optimization problem and allows vertices to belong to more than one cluster, to detect interesting sub-graphs in the attributed graph. Then, we propose EGCPI, which is an evolutionary algorithm for cluster detection in the attributed graph. EGCPI tackles the problem of cluster detection with evolutionary clustering. It can identify those sub-graphs in which vertices are densely connected, as well as the attributes of vertices, are more similar. At last, we propose TBPCI, which is able to identify clusters in the attributed graph utilizing the local information on the vertex connectedness and associated attributes values. By identifying the optimal degree of boundedness between pairwise vertices and grouping those vertices sharing higher boundedness, TBPCI may discover clusters in which vertices are densely connected and their attribute values are significantly associated.

To show the effectiveness and efficiency of these proposed algorithms, we have tested them with different types of attributed graph data, including synthetic graphs, social

graphs, and biological graphs, and compared them with the state-of-the-art approaches to graph clustering. These proposed algorithms outperform most state-of-the-art approaches in most of the testing datasets, according to the experimental results related to different evaluation metrics, such as overall accuracy and efficiency. These proposed algorithms may well address the challenges mentioned in the thesis and they are effective for mining meaningful clusters in the attributed graph.

8.2. Future work

In future, we attempt to improve the proposed approaches from the following aspects.

As the size of graph data is increasing tremendously, traditional algorithms for mining clusters might not be capable of the intractable task. Based on the current approaches proposed, we attempt to develop several computationally efficient algorithms for discovering clusters in very big attributed graphs. Specifically, we will propose more efficient updating methods for those approaches based on iterative optimization, i.e., MISAGA, FSPGA, and TBPCI. We will try to propose parallel methods for sub-space search for those algorithms based on natural computation, i.e., EGCPI. Moreover, we attempt to propose incremental versions of the proposed algorithms to deal with the real-world problem, e.g., clustering in streaming attributed graph data.

As the proposed algorithms mainly consider those objective functions evaluating the overall quality of the discovered clusters rather than find clusters based on the graph hierarchy, we attempt to propose novel versions of the proposed algorithms which may discover clusters taking into the consideration both topological and attribute hierarchy hidden in the attributed graph.

Besides identifying clusters with vertices which are topologically and characteristically interrelated, approaches that can search for the information representing the detected

clusters within the clustering procedure, might be desirable. Apparently, obtaining such representative information may better explain the meaning of the found clusters in the attributed graph. Hence, we aim to develop more methods that can discover clusters in the attributed graph by taking into the consideration latent factors that may represent the meaning of each cluster.

Given the features of the proposed algorithms and the state-of-the-art, how to identify the interdependency between graph topology and attribute values in the attributed graph is a challenging task. Rather than concerning both as the combinatorial factors within the clustering procedure, we attempt to propose adaptive methods that may automatically determine the dominant factors taking effect on the cluster membership, which might tremendously improve the efficiency and effectiveness of detecting clusters in the attributed graph.

As the current version of the proposed algorithms mainly try to find clusters in a static graph, we attempt to develop extended versions of the proposed algorithms to discover clusters in dynamical graph data, which contain a series of graphs changing along with the time stamp. Such extensions may significantly improve the impact of the proposed algorithms.

REFERENCES

[1] Y. Ahn, J. P. Bagrow, and S. Lehmann, "Link communities reveal multiscale complexity in networks," *Nature*, vol. 466, pp. 761-764, Aug. 2010.

[2] E. M. Airoldi, et al., "Mixed Membership Stochastic Blockmodels," *J. Mach. Learn. Res.*, vol. 9, pp. 1981-2014, Sep. 2008.

[3] M. Altaf-Ul-Amin, Y. Shinbo, K. Mihara, K. Kurokawa, and S. Kanaya, "Development and Implementation of an Algorithm for Detection of Protein Complexes in Large Interaction Networks," *BMC Bioinformatics*, vol. 7, no. 1, article 207, 2006.

[4] B. Adamcsek, G. Palla, I. J. Farkas, I. Derenyi, and T. Vicsek, "CFinder: Locating Cliques and Overlapping Modules in Biological Networks," *Bioinformatics*, vol. 22, no. 8, pp. 1021-1023, 2006.

[5] M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig, M. A. Harris, D. P. Hill, L. Issel-Tarver, A. Kasarskis, S. Lewis, J. C. Matese, J. E. Richardson, M. Ringwald, G. M. Rubin, and G. Sherlock, "Gene Ontology: Tool for the Unification of Biology," *Nature Genetics*, vol. 25, no. 1, pp. 25-29, 2000.

[6] D. T. Anderson, A. Zare, and S. Price, "Comparing Fuzzy, Probabilistic, and Possibilistic Partitions Using the Earth Mover's Distance," *IEEE Trans. Fuzzy Syst.*, vol. 21, no. 4, pp. 766-775, Aug. 2013.

[7] D. T. Anderson, J. C. Bezdek, M. Popescu, and J. M. Keller, "Comparing Fuzzy, Probabilistic, and Possibilistic Partitions," *IEEE Trans. Fuzzy Syst.*, vol. 18, no. 5, pp. 906-918, Oct. 2013.

[8] V. D. Blondel, et al., "Fast unfolding of communities in large networks," *J. Stat. Mech.*, vol. 2008, no. 10, pp. P10008, Oct. 2008.

[9] J. C. Bezdek, R. Ehrlich, W. Full, "FCM: the fuzzy c-means clustering algorithm," *Computers and Geosciences*, vol.10, no. 2-3, pp. 191-203, 1984.

[10] D. Blei, "Probabilistic topic models," *Commun. ACM*, vol. 55, no. 4, pp. 77-84, 2012.

[11] R. Balasubramanyan, and W. W. Cohen, "Block-LDA: Jointly modeling entity-annotated text and entity-entity links," in *Proc. SIAM Int. Conf. Data Mining*, 2011, pp. 450-461.

[12] S. Bechikh, A. Chaabani, and L. B. Said, "An efficient chemical reaction optimization algorithm for multiobjective optimization," *IEEE Trans. Cybern.*, vol. 45, no. 10, pp. 2051-2064, Oct. 2015.

[13] G. Bader, and C. Hogue, "An Automated Method for Finding Molecular Complexes in Large Protein Interaction Networks," *BMC Bioinformatics*, vol. 4, article 2, 2003.

[14] T. Bäck, *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*, 1st ed. Oxford, UK: Oxford Univ. Press, 1996.

[15] E. I. Boyle, S. Weng, J. Gollub, H. Jin, D. Botstein, J. M. Cherry, and G. Sherlock, "GO::TermFinder—Open Source Software for Accessing Gene Ontology Information and Finding Significantly Enriched Gene Ontology Terms Associated with a List of Genes," *Bioinformatics*, vol. 20, no. 18, pp. 3710-3715, 2004.

[16] A. Bertoni, and G. Valentini, "Model order selection for bio-molecular data clustering," *BMC Bioinformatics*, vol. 8, no. 2, pp. S7, 2007.

[17] A. Clauset, M. E. J. Newman, and C. Moore, "Finding community structure in very large networks," *Phys. Rev. E*, vol. 70, no. 6, pp. 066111, Dec. 2004.

[18] J. Chan, and D. Blei, "Relational topic models for document networks," in *Proc.*

12th Int. Conf. Artificial Intel. Stat., 2009, pp. 81-88.

[19] H. Chang, Z. Feng, and Z. Ren, "Community detection using dual-representation chemical reaction optimization," *IEEE Trans. Cybern.*, vol. PP, no. 99, pp. 1-14, 2016.

[20] K. C. C. Chan, A. K. C. Wong, and D. K. Y. Chiu, "Learning sequential patterns for probabilistic inductive prediction," *IEEE Trans. Systems, man and cybernetics*, vol. 24, no. 10, pp. 1532-1547, Oct. 1994.

[21] J. Y. Ching, A. K. C. Wong, and K. C. C. Chan, "Class-dependent discretization for inductive learning from continuous and mixed-mode data," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 17, no. 7, pp. 641-651, Jul. 1995.

[22] I. Chiang, C. C. Liu, Y. Tsai, and A. Kumar, "Discovering latent semantics in web documents using fuzzy clustering," *IEEE Trans. Fuzzy Systems*, vol. 23, no. 6, pp. 2122-2134, Dec. 2015.

[23] S. R. Collins, P. Kemmeren, X. Zhan, J. F. Greenblatt, F. Spencer, F. C. P. Holstege, J. S. Weissman, and N. J. Krogan, "Toward a comprehensive atlas of the physical interactome of saccharomyces cerevisiae," *Molecular&Cellular Proteomics*, vol. 6, pp. 439-450, Mar. 2007.

[24] E. Camon, M. E. Camon, M. Magrane, D. Barrell, V. Lee, E. Dimmer, J. Maslen, D. Binns, N. Harte, R. Lopez, and R. Apweiler, "The Gene Ontology Annotation (GOA) Database: Sharin Knowledge in Uniprot with Gene Ontology," *Nucleic Acids Research*, vol. 32, no. Suppl. 1, pp. D262-D266, 2003.

[25] J. M. Cherry, C. Adler, C. Ball, S. A. Chervitz, S. S. Dwight, E. T. Hester, Y. Jia, G. Juvik, T. Roe, M. Schroeder, S. Weng, and D. Botstein, "SGD: Saccharomyces Genome Database," *Nucleic Acids Research*, vol. 26, no. 1, pp. 73-79, 1998.

[26] F. M. Couto, M. J. Silva, and P. M. Coutinho, "Measuring Semantic Similarity

between Gene Ontology Terms," *Data Knowledge Eng.*, vol. 61, no. 1, pp. 137-152, 2007.

[27] F. M. Couto, M. Silva, and P. M. Coutinho, "Semantic Similarity over the Gene Ontology: Family Correlation and Selecting Disjunctive Ancestors," in *Proc. ACM Int'l Conf. Information and Knowledge Management (CIKM '05)*, pp. 343-344, 2005.

[28] T. Deisboeck, and J. Y. Kresh, *Complex systems Science in BioMedicine*, Springer, 2006.

[29] S. v. Dongen, "Graph Clustering by Flow Simulation," PhD thesis, Univ. of Utrecht, The Netherlands, 2000.

[30] S. v. Dongen, "A Cluster Algorithm for Graphs," *Technical Report*, R 0010, CWI, 2000.

[31] X. Ding, W. Wang, X. Peng, and J. Wang, "Mining Protein Complexes from PPI Networks Using the Minimum Vertex Cut," *Tsinghua Science and Technology*, vol. 17, no. 6, pp. 674-681, 2012.

[32] A. Dempster, N. Laird, and D. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of Royal Statistics Society*, vol. 39, pp. 1-38, 1977.

[33] S. Fortunato, "Community detection in graphs," *Phys. Rep.*, vol. 486, no.3-5, pp. 75-174, Feb. 2010.

[34] S. Fortunato, and M. Barthelemy, "Resolution limit in community detection," *Proc. Nat. Acad. Sci. U.S.A.*, vol. 104, no. 1, pp. 36-41, Jan. 2007.

[35] B. J. Frey, and D. Dueck, "Clustering by passing messages between data points," *Science*, vol. 16, pp. 972-976, Feb. 2007.

[36] M. Frank, et al., "Multi-assignment clustering for boolean data," *J. Mach. Learn.*

Res., vol. 13, pp. 459-489, Feb. 2012.

[37] A. Gavin, P. Aloy, P. Grandi, R. Krause, M. Boesche, M. Marzioch, C. Rau, L.J. Jensen, S. Bastuck, B. Dumpelfeld, A. Edelmann, M. Heurtier, V. Hoffman, C. Hoefert, K. Klein, M. Hudak, A. Michon, M. Schelder, M. Schirle, M. Remor, T. Rudi, S. Hooper, A. Bauer, T. Bouwmeester, G. Casari, G. Drewes, G. Neubauer, J.M. Rick, B. Kuster, P. Bork, R.B. Russell, and G. Superti-Furga, "Proteome survey reveals modularity of the yeast of the yeast cell machinery," *Nature*, vol. 440, pp. 631-636, 2006.

[38] A. Gavin, M. Bosche, R. Krause, P. Grandi, M. Marzioch, A. Bauer, J. Schultz, J. M. Rick, A. Michon, C. Cruciat, M. Remor, C. Hofert, M. Schelder, M. Brajenovic, H. Ruffner, A. Merino, K. Klein, M. Hudak, D. Dickson, T. Rudi, V. Gnau, A. Bauch, S. Bastuck, B. Huhse, C. Leutwein, M. Heurtier, R. R. Copley, A. Edelmann, E. Querfurth, V. Rybin, G. Drewes, M. Raida, T. Bouwmeester, P. Bork, B. Seraphin, B. Kuster, G. Newbauer, and G. Superti-Furga, "Functional Organization of the Yeast Proteome by Systematic Analysis of Protein Complexes," *Nature*, vol. 415, pp. 141-147, 2002.

[39] M. Girvan, and M. E. J. Newman, "Community structure in social and biological networks," *Proc. Nat. Acad. Sci. U. S. A.*, vol. 99, no. 12, pp. 7821-7826, Jun. 2002.

[40] R. Guimera, M. Sales-Pardo, and L. A. N. Amaral, "Modularity from fluctuations in random graphs and complex networks," *Phys. Rev. E*, vol. 70, no. 2, pp. 025101, Aug. 2004.

[41] S. Gunnermann, et al., "Efficient mining of combined subspace and subgraph clusters in graphs with feature vectors," in *Proc. PAKDD*, 2013, pp.261-275.

[42] U. Güldener, M. Münsterkötter, G. Kastenmüller, N. Strack, J. van Helden, C. Lemer, J. Richelles, S. J. Wodak, J. García-Martínez, J. E. Pérez-Ortín, H. Michael,

A. Kaps, E. Talla, B. Dujon, B. André, J. L. Souciet, J. De Montigny, E. Bon, C. Gaillardin, and H. W. Mewes, "CYGD: The Comprehensive Yeast Genome Database," *Nucleic Acids Research*, vol. 33, no. suppl. 1, pp. D364-D368, 2005.

[43] R. J. Hathaway, J. W. Davenport, and J. C. Bezdek, "Relational duals of the c-means clustering algorithms," *Pattern Recog.*, vol. 22, no. 2, pp.205-212, 1989.

[44] L. Hu and K. C. C. Chan, "Fuzzy Clustering in a Complex Network Based on Content Relevance and Link Structures," *IEEE Trans. Fuzzy Systems*, vol. 24, no. 2, pp. 456-470, Apr. 2016.

[45] T. He, and K. C. C. Chan, "Evolutionary community detection in social networks," in *Proc. CEC*, 2014, pp. 1496-1503.

[46] Y. Ho, A. Gruhler, A. Heilbut, G. D. Bader, L. Moore, S. Adams, A. Millar, P. Taylor, K. Bennett, K. Boutilier, L. Yang, C. Wolting, I. Donaldson, S. Schandorff, J. Shewnarane, M. Vo, J. Taggart, M. Goudreault, B. Muskat, C. Alfarano, D. Dewar, Z. Lin, K. Michalickova, A. R. Willems, H. Sassi, P. A. Nielsen, K. J. Rasmussen, J. R. Andersen, L. E. Johansen, L. H. Hansen, H. Jespersen, A. Podtelejnikov, E. Nielsen, J. Crawford, V. Poulsen, B. D. Sorensen, J. Matthiesen, R.C. Hendrickson, F. Gleeson, T. Pawson, M. F. Moran, D. Durocher, M. Mann, C. W. V. Hogue, D. Figeys, and M. Tyers, "Systematic Identification of Protein Complexes in Saccharomyces cerevisiae by Mass Spectrometry," *Nature*, vol. 415, pp. 180-183, 2002.

[47] L. Hu, and K. C. C. Chan, "Utilizing both topological and attribute information for protein complex identification in PPI networks," *IEEE/ACM Trans. Computational Biology and Bioinformatics*, vol. 10(3), pp 780-792, May 2013.

[48] S. J. Haberman, "The Analysis of Residuals in Cross-Classified Tables," *Biometrics*, vol. 29, pp. 205-220, 1973.

[49] R. J. Hathaway, J. C., Bezdek, “Switching regression models and fuzzy clustering,” *IEEE Trans. Fuzzy Systems*, vol. 1, no. 3, pp. 195-204, 1993.

[50] Y. Hong, S. Kwong, H. Xiong, and Q. Ren, “Genetic-guided semi-supervised clustering algorithm with instance-level constraints,” in *Proc. 10th Annu. Conf. Genetic and evolutionary computation*, 2008, pp. 1381–1388.

[51] J. Ji, A. Zhang, C. Liu, and X. Quan, “Survey: Functional module detection from protein-protein interaction networks,” *IEEE Trans. Knowledge and Data Engineering*, vol. 26, no. 2, pp. 261-277, 2014.

[52] P. Jiang, and M. Singh, “SPICi: a fast clustering algorithm for large biological networks,” *Bioinformatics*, vol. 26, no. 8, pp. 1105-1111, Apr. 2010.

[53] B. Karrer, and M. E. J. Newman, “Stochasitc blockmodels and community structure in networks,” *Phys. Rev. E*, vol. 83, no. 1, pp. 016107, Jan. 2011.

[54] M. A. Khalilia, J. Bezdek, M. Popescu, and J. M. Keller, “Improvements to the relational fuzzy c-means clustering algorithm,” *Pattern Recog.*, vol. 47, no. 12, pp.3920-3930, 2014.

[55] A. D. King, N. Przulj, and I. Jurisica, “Protein Complex Prediction via Cost-Based Clustering,” *Bioinformatics*, vol. 20, no. 17, pp. 3013-3020, 2004.

[56] N. J. Krogan, et al., “Global Landscape of Protein Complexes in the Yeast Saccharomyces cerevisiae,” *Nature*, vol. 440, no. 7084, pp. 637-643, 2006.

[57] U. Luxburg, “A tutorial on spectral clustering,” *Statistics and Computing*, vol. 17, no. 4, pp. 395-426, Dec. 2007.

[58] J. Leskovec, K. J. Lang, and M. Mahoney, “Empirical comparison of algorithms for network community detection,” in *Proc 19th Int. conf. World Wide Web*, 2010, pp. 631-640.

[59] M. Lipczak, and E. Milios, "Agglomerative genetic algorithm for clustering in social networks," in *Proc. 11th Annu. Conf. Genetic and evolutionary computation*, 2009, pp. 361-362.

[60] A. Lancichinetti, S. Fortunato, and J. Kertesz, "Detecting the overlapping and hierarchical community structure in complex networks," *New Journal of Physics*, vol. 11, no. 3, pp. 033015, Mar. 2009.

[61] C. Liu, J. Liu, and Z. Jiang, "A multiobjective evolutionary algorithm based on similarity for community detection from signed social networks," *IEEE Trans. Cybern.*, vol. 44, no. 12, pp. 2274-2287, Dec. 2014.

[62] X. Li, M. Wu, C. K. Kwoh, and S. K. Ng, "Computational approaches for detecting protein complexes from protein interaction networks: a survey," *BMC Genomics*, vol. 11(1) article 1, 2010.

[63] M. Li, J. Chen, J. Wang, B. Hu, and G. Chen, "Modifying the DPClus Algorithm for Identifying Protein Complexes Based on New Topological Structures," *BMC Bioinformatics*, vol. 9, no. 1, article 398, 2008.

[64] G. Liu, L. Wong, and H. N. Chua, "Complex Discovery from Weighted PPI Networks," *Bioinformatics*, vol. 25, no. 15, pp. 1891-1897, 2009.

[65] M. Li, J. Wang, J. Chen, and Z. Cai, "Identifying the overlapping complexes in protein interaction networks," *International Journal of Data Mining and Bioinformatics*, vol. 4, no.1, pp. 91-108, 2010.

[66] W. W. M. Lam, and K. C. C. Chan, "Discovering Functional Interdependence Relationship in PPI Networks for Protein Complex Identification," *IEEE Trans. Biomedical Eng.*, vol. 59, no. 4, pp. 899-908, Apr. 2012.

[67] M. Li, X. Wu, J. Wang, and Y. Pan, "Towards the Identification of Protein Complexes and Functional Modules by Integrating PPI Network and Gene

Expression Data," *BMC Bioinformatics*, vol. 13, no. 1, article 109, 2012.

[68] G. Liu, C. H. Yong, H. N. Chua, and L. Wong, "Decomposing PPI Networks for Complex Discovery," *Proteome Science*, vol. 9, no. Suppl. 1, article S15, 2011.

[69] M. Li, T. Yu, X. Wu, J. Wang, F. Wu, and Y. Pan, "C-DEVA: detection, evaluation, visualization and annotation of clusters from biological networks," *BioSystems*, vol. 150, pp. 78-86, 2016.

[70] W. Lord, R. D. Stevens, A. Brass, and C. A. Goble, "Semantic Similarity Measures as Tools for Exploring the Gene Ontology," in *Proc. Pacific Symp. Biocomputing*, 2003, pp. 601.

[71] Z. Lei and Y. Dai, "Assessing Protein Similarity with Gene Ontology and Its Use in Subnuclear Localization Prediction," *BMC Bioinformatics*, vol. 7, no. 1, article 491, 2006.

[72] U. V. Luxburg, *Clustering Stability*, Now Publishers Inc., 2010.

[73] D. J. C. MacKay, *Information Theory, Inference, and Learning Algorithms, Cambridge*, UK: Cambridge Univ. Press, 2003.

[74] J. McAuley, and J. Leskovec, "Discovering social circles in ego networks," *ACM Trans. Knowl. Discov. Data*, vol. 8, no. 1, article 4, 2014.

[75] H. W. Mewes, D. Frishman, U. Guldener, G. Mannhaupt, K. Mayer, M. Mokrejs, B. Morgenstern, M. Munsterkotter, S. Rudd, and B. Weil, "MIPS: A Database for Genomes and Protein Sequences," *Nucleic Acids Research*, vol. 30, no. 1, pp. 31-34, 2002.

[76] M. E. J. Newman, "Modularity and community structure in networks," *Proc. Nat. Acad. Sci. U. S. A.*, vol. 103, no. 23, pp. 8577-8582, Jun. 2006.

[77] R. Nallapati, et al., "Joint topic models for text and citations," in *Proc. 14th ACM*

Int. Conf. Kowl. Discov. Data Mining, 2008, pp. 542-550.

[78] T. Nepusz, H. Yu, and A. Paccanaro, "Detecting overlapping protein complexes in protein-protein interaction networks,", *Nat. Methods*, vol. 9, pp. 471-472, 2012.

[79] T. M. Nguyen, and Q. M. J. Wu, "Dynamic fuzzy clustering and its application in motion segmentation," *IEEE Trans. Fuzzy Systems*, vol. 21, no. 6, pp. 1019-1031, Dec. 2013.

[80] G. Palla, et al., "Uncovering overlapping community structure of complex networks in nature and society," *Nature*, vol. 435, pp. 814-818, Jun. 2005.

[81] C. Pizzuti, "A multiobjective genetic algorithm to find communities in complex networks," *IEEE Trans. Evol. Comput.*, vol. 16, no. 3, pp. 418-430, Jun. 2012.

[82] W. Peng, J. Wang, B. Zhao, and L. Wang, "Identification of protein complexes using weighted pagerank-nibble algorithm and core-attachment structure," *IEEE/ACM Trans. Computational Biology and Bioinformatics*, vol. 12, no. 1, pp. 179-192, Jan./Feb. 2015.

[83] S. Pu, et al., "Up-to-date catalogues of yeast protein complexes," *Nucleic Acids Res.*, vol. 37, no. 3, pp. 825-831, Feb. 2009.

[84] N. R. Pal, K. Pal, and J. M. Keller, "A possibilistic fuzzy c-means clustering algorithm," *IEEE Trans. Fuzzy Systems*, vol. 13, no. 4, pp. 517-530, 2005.

[85] G. Qi, C. C. Aggarwal, and T. Huang, "Community detection with edge content in social media networks," in *Proc. IEEE 28th Int. Conf. Data Engineering*, 2012, pp. 534-545.

[86] P. J. Rousseeuw, "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis," *J. Computational Appl. Math.*, vol. 20, pp. 53-65, Nov. 1987.

[87] J. Ren, J. Wang, M. Li, and L. Wang, "Identifying protein complexes based on

density and modularity in protein-protein interaction network," *BMC Systems Biology*, vol. 7, no. 4, article 1, 2013.

[88] A. Ruepp, B. Brauner, I. Dunger-Kaltenbach, G. Frishman, C. Montrone, M. Stransky, B. Waegele, T. Schmidt, O. N. Doudieu, V. Stümpflen, and H. W. Mewes, "CORUM: The Comprehensive Resource of Mammalian Protein Complexes," *Nucleic Acids Research*, vol. 36, no. suppl. 1, pp. D646-D650, 2008.

[89] J. Shi, and J. Malik, "Normalized cuts and image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8, pp. 888-905, Aug. 2000.

[90] Y. Sun, et al., "itopicmodel: information network-integrated topic modeling," in *Proc. IEEE 9th Int. Conf. Data Mining*, 2009, pp. 493-502.

[91] V. Spirin, and L. A. Mirny, "Protein Complexes and Functional Modules in Molecular Network," *Proc. Nat'l Academy of Sciences, USA*, vol. 100, no. 21, pp. 12123-12128, 2003.

[92] L. Silva, R. Moura, A. M. P. Canuto, R. H. N. Santiago, and B. Bedregal, "An interval based frame work for fuzzy clustering applications," *IEEE Trans. Fuzzy Systems*, vol. 23, no. 6, pp. 2174-2187, Mar. 2015.

[93] C. Stark, B. Breitkreutz, T. Reguly, L. Boucher, A. Breitkreutz, and M. Tyers, "BioGRID: A General Repository for Interaction Datasets," *Nucleic Acids Research*, vol. 34, no. Suppl. 1, pp. D535-D539, 2006.

[94] M. Seeland, T. Girschick, F. Buchwald, "Online structural graph clustering using frequent sub-graph mining," in *Proc. ECML-PKDD*, 2010, pp. 213-228.

[95] Y. Tian, R. A. Hankins, and J. M. Patel, "Efficient aggregation for graph summarization," in *Proc. of the 2008 ACM Int. Conf. Management of Data*, pp. 567-580, 2008.

[96] M. Tasgin, A. Herdagdelen, and H. Bingol, "Community detection in complex

networks using genetic algorithms," in *Proc. Eur. Conf. Complex Systems*, 2006, pp. 57.

[97] A. H. Y. Tong, B. Drees, G. Nardelli, G. D. Bader, B. Brannetti, L. Castagnoli, M. Evangelista, S. Ferracuti, B. Nelson, S. Paoluzi, M. Quondam, A. Zucconi, C. W. V. Hogue, S. Fields, C. Boone, and G. Cesareni, "A combined Experimental and Computational Strategy to Define Protein Interaction Networks for Peptide Recognition Modules," *Science*, vol. 295, no. 5553, pp. 321-324, 2002.

[98] A. L. Traud, et al., "Comparing community structure to characteristics in online collegiate social networks," *SIAM Rev.*, vol. 53, no. 3, pp. 526-543, Aug. 2011.

[99] G. Valentini, "Clusterv: a tool for assessing the reliability of clusters discovered in DNA microarray data," *Bioinformatics*, vol. 22, no. 3, pp. 369-370, 2006.

[100] M. Wu, X. Li, C. Kwoh, and S. Ng, "A Core-Attachment Based Method to Detect Protein Complexes in PPI Networks," *BMC Bioinformatics*, vol. 10, no. 1, article 169, 2009.

[101] Y. Wang, R. Wang, X. Zhang, and L. Chen, "Establishing protein functional linkage in a systematic way," *Lect. Notes Oper. Res.*, vol. 7, pp. 75-88, 2007.

[102] C. H. Wu, R. Apweiler, A. Bairoch, D. A. Natale, W. C. Barker, B. Boeckmann, S. Ferr, E. Gasteiger, H. Huang, R. Lopez, M. Magrane, M. J. Martin, R. Mazumder, C. O'Donovan, N. Redaschi, and B. Suzek, "The universal protein resource (UniProt): an expanding universe of protein information," *Nucleic acids research*, vol. 34, D187-D191, 2006.

[103] J. Wang, G. Chen, B. Liu, M. Li, and Y. Pan, "Identifying Protein Complexes from Interactome Based on Essential Proteins and Local Fitness Method," IEEE Trans. NanoBioscience, vol. 11, no. 4, pp. 324-335, Dec. 2012.

[104] Z. Xu, et al., "Gbagc: a general bayesian framework for attributed graph

clustering," *ACM Trans. Knowl. Discov. Data*, vol. 9, no. 1, article 5, 2014.

[105] I. Xenarios, et al., "DIP, the Database of Interacting Proteins: A Research Tool for Studying Cellular Networks of Protein Interactions," *Nucleic Acids Research*, vol. 30, no. 1, pp. 303-305, 2002.

[106] L. Xie, and G. Beni, "A validity measure for fuzzy clustering," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 13, no. 8, pp. 841-847, Aug. 1991.

[107] L. Yang, et al., "A unified semi-supervised community detection framework using latent space graph regularization," *IEEE Trans. Cybern.*, vol. 45, no. 11, pp. 2585-2598, Nov. 2015.

[108] J. Yang, J. McAuley, and J. Leskovec, "Detecting Cohesive and 2-mode Communities in Directed and Undirected Networks," in *Proc. 7th ACM Int. Conf. Web Search and Data Mining*, 2014, pp. 323-332.

[109] T. Yang, et al., "Combining link and content for community detection: a discriminative approach," in *Proc. 15th ACM Int. Conf. Kowl. Discov. Data Mining*, 2009, pp. 927-936.

[110] J. Yang, J. McAuley, and J. Leskovec, "Community detection in networks with node attributes," in *Proc. IEEE 13th Int. Conf. Data Mining*, 2013, pp. 1151-1156.

[111] X. Yan, and J. Han, "gspan: Graph-based substructure pattern mining," in *Proc. IEEE 2nd Int. Conf. Data Mining*, 2002, pp. 721-724.

[112] Y. Zhou, H. Cheng, and J. X. Yu, "Graph clustering based on structural/attribute similarities," in *Proc. VLDB*, 2009, pp. 718-729.

[113] Y. Zhou, H. Cheng, and J. X. Yu, "Clustering large attributed graphs: an efficient incremental approach," in *Proc. IEEE 10th Int. Conf. Data Mining*, 2010, pp. 689-698.

[114] X. Zhang, R. Wang, Y. Wang, and J. Wang, "Modularity optimization in community detection of complex networks," *Europhysics Letters*, vol. 87, no. 3, pp. 38002, 2009.

[115] B. Zhao, J. Wang, M. Li, F. Wu, and Y. Pan, "Detecting protein complexes based on uncertain graph model," *IEEE/ACM Trans. Computational Biology and Bioinformatics*, vol. 11, no. 3, pp. 486-497, May/Jun. 2014.

[116] X. Zhang, D. Dai, and X. Li, "Protein Complexes Discovery Based on Protein-Protein Interaction Data via a Regularized Sparse Generative Network Model," *IEEE/ACM Trans. Computational Biology and Bioinformatics*, vol. 9, no. 3, pp. 857-870, May/June 2012.

[117] X. Zhang, D. Dai, L. Ouyang, and H. Yan, "Detecting overlapping protein complexes based on a generative model with functional and topological properties," *BMC Bioinformatics*, vol. 15, article 186, 2014.