MFC: Mining Maximal Frequent Dense Subgraphs without Candidate Maintenance in Imbalanced PPI Networks

Miao Wang[1], Xuequn Shang[1, §], Zhanhuai Li[1]
[1]School of Computer Science and Engineering, Northwestern Polytechnical University, Xi'an, China
[§]Corresponding author, E-mail addresses: shang@nwpu.edu.cn

***Abstract*—The prediction of protein function is one of the most challenging problems in bioinformatics. Several studies have shown that the prediction using PPI is promising. However, the PPI data generated from high-throughput experiments are very noisy, which renders great challenges to the existing methods. In this paper, we propose an algorithm, MFC, to efficiently mine maximal frequent dense subgraphs without candidate maintenance in PPI networks. Instead of using summary graph, MFC produces frequent dense patterns by extending vertices. It adopts several techniques to achieve efficient mining. Due to the imbalance character of PPI network, we also propose to generate frequent patterns using relative support. We evaluate our approach on four PPI data sets. The experimental results show that our approach has good performance in terms of efficiency. With the help of relative support, more frequent dense functional interaction patterns in the PPI networks can be identified.**

***Index Terms*—frequent dense subgraph, imbalance, relative support, used edge, family subgraph**

I. INTRODUCTION

Functional annotation of protein is one of the fundamental problems in the post-genomic era. The widely used methods are sequence alignment[1-3] and pattern-discovery[4-6]. However, these methods have their own inherent drawbacks[7]. The recent development of high-throughout technology for protein-protein interaction (PPI) measurements[8,9] have generated large-scale data on protein interaction across human and other organisms, which are expected to be fertile sources of information for deriving protein functions.

Prediction of protein functions using PPI data has been extensively studied. These methods include neighborhood based[11], graph theoretic[12], probabilistic[13], topology[14], expanding seed complex[15], integrating multiple data sources[16], etc. One commonly used analytical approach is to discover the clusters[17-20], which is likely to share the same functions. It is observed that 70-80% of proteins share at least one function with its interacting partner[21]. However, due to the noisy nature of high-throughput data, the above assumption is not always true. One way to overcome it is to mine dense frequent patterns in multiple biological networks simultaneously. Recently, many algorithms have been proposed. Hu et, and Yan et, [22,23] exploited summary graph approach to mine frequent patterns. But it may suffer from several problems. (1) The noise may aggregate to affect the mining efficiency[23]. (2) The edges in a summary graph may not be dense. Pruning the non-dense edges is time-consuming. (3) Mining imbalanced networks is not efficient since the summary graph may be large when dealing with imbalanced networks. When generating second-order graph, the computing could be complex. (4) The edges in a dense summary graph may never occur together in individual original graphs[23], which can make summary graph larger. Fig. 1(a) illustrates such an example with a cartoon of four graphs. If we simply add these graphs together to get a summary graph, we may find a dense subgraph with vertices *a*, *b*, *c* and *d*. However, this subgraph is not frequent in the original graphs. So these algorithms use other techniques to overcome these problems. Since summary graph should be produced first, so if the summary graph is not effective, it must affect the efficiency of later technique.

In this paper, we address the above issues and develop an algorithm, called 'MFC', the abbreviation of mining Maximal Frequent dense subgraphs without Candidate maintenance in imbalanced PPI datasets. MFC adopts the vertex-growth method, in which one iteration may generate more frequent edges. In addition, it exploits several pruning techniques to avoid storing the generated maximal frequent dense subgraphs and prune unmaximal subgraphs in time. In comparison with previous frequent graph mining algorithms, our algorithm may show significant advantage in memory and computing efficiency.

The PPI datasets are commonly represented as relational graph[10], where nodes representing unique proteins and edges representing the unique relationship between proteins. That means, relational graph has distinct node labels, and we do not have the 'subgraph isomorphism problem' which is NP-hard. Instead of using summary graph and growing patterns by extending the edges, we mine frequent patterns by extending the vertices. Using traditional methods mine frequent dense subgraph, it should produce frequent subgraph firstly, then judging which is whether dense or not. If not, it may use other technologies to separate the subgraph into several dense subgraphs. So the efficiency is not very

doi:10.4304/jsw.6.3.498-507

well. Using MFC, the frequent dense subgraphs can be generated at the same time. One chosen vertex must satisfy the condition: the new produced patterns must be frequent and dense. We propose a new dense definition: similar density. Biologically speaking, if one protein is more likely to be in the module, it must connect to more proteins in this module. If the similar degree is lower, it can not be joined in this module. At the same time, the module must be found in most original graphs. Using our method can avoid mining the false patterns. MFC can also discover the overlapping graph clusters. As we known, one protein may have different functions, so identifying overlapping clusters is important in biological applications. For example, in Fig. 1 (b), two cliques $\{a,b,c,d,e,f\}$ and $\{d,e,g,h\}$ share two common vertices $\{d,e\}$. MFC can identify these two clusters easily. The detail can be found in the method part.

As we known, different PPI dataset is got from different experiment, due to the influence of environment, time or other factors, one protein is discovered in one experiment may not be found in another experiment. Therefore, using traditional support to discover functional module may ignore some important modules. In a word, imbalance and perturbation are two important characters of PPI network. So we propose another definition: relative support, which aims at finding more patterns in imbalanced networks. Using relative support can discover more frequent subgraphs than absolute support. We applied our algorithm to 4 human PPI datasets to identify a large number of potential function modules. Assessed by Gene Ontology, the results show MFC is efficient to discover annotated modules. The experimental results also show using relative support can discover more frequent subgraphs than traditional absolute support.

The rest of the paper is organized as follows: In Section 2, we present the problem definition of mining maximal frequent dense subgraphs. Section 3 is focus on the MFC algorithm, mainly including how MFC gets maximal frequent dense subgraphs without candidate maintenance. In Section 4, we present an extensive experimental study. Our study is concluded in Section 5.

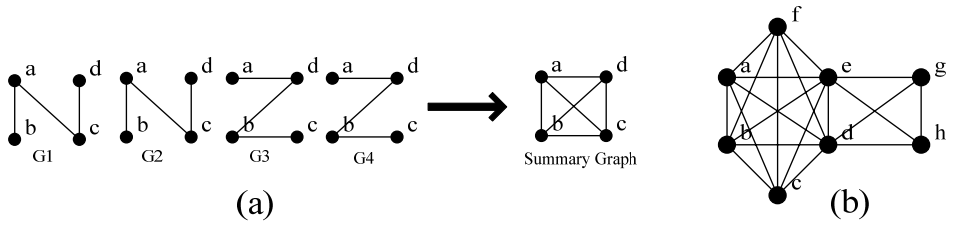

Figure 1(a) The summary graph of four graphs. (b) The overlapping dense subgraphs.

II. PROBLEM FORMULATION

A relational graph set consists of undirected simple graphs, $D=\{G_i=(V,E_i)\}$, $i=1,\cdots,n$, $E_i \subseteq V \times V$, while a common vertex V is shared by the graphs in the set. We denote the vertex set of a graph G by $V(G)$ and the edge set by $E(G)$. In relational graphs, there is neither loop nor multiple edges. All the graph patterns discussed in this paper are undirected connected relational graphs.

Definition 1 (Absolute Support). Given a relation graph dataset, $D=\{G_1,G_2,\cdots,G_n\}$, while $G_i=(V,E_i)$. The number of graphs in D where g is a subgraph is r, the absolute support of a graph g is r/n, written $ASup(g)$. A subgraph is frequent if its support is greater than a minimum support threshold.

Definition 2 (Vertex Support). Given a relational graph dataset, $D=\{G_1,G_2,\cdots,G_n\}$, while $G_i=(V,E_i)$. The number of graphs in D which contain v is r, the vertex support of a vertex v is r/n, written $VSup(v)$.

Definition 3 (Relative Support). Given a relational graph dataset, $D=\{G_1,G_2,\cdots,G_n\}$, while $G_i=(V,E_i)$, r is the number of graphs in D which contain a subgraph g, $V(g)=\{v_1,v_2,\cdots v_i\}$, the relative support of g is r/T, where $T=min\{VSup(v_1),VSup(v_2),\cdots VSup(v_i)\}\times n$, written $RSup(g)$. g is frequent if its relative support is greater than a minimum support threshold.

E.g. Given a subgraph g:$\{ae,ce,ab\}$, in Fig. 2, the support threshold is 0.6. $ASup(g)$=2/4=0.5<0.6, so it is not frequent. However, using relative support to mine, since the vertex e can not be found only in G3, $RSup(g)$=2/3=0.67>0.6, it is frequent. So using relative support we can discover more frequent patterns.

However, only using relative support can mine some noise patterns. As shown in Fig. 2, the edge af only can be found in $G3$, if using relative support, $RSup(af)$=1, but it may be a noise edge. Therefore, before using relative support, absolute support must be used to reduce the noise vertices and edges. Certainly, the absolute support should be smaller than relative support. In above example, if we set the absolute support is 0.5, the noise edge af would be cut firstly.

Definition 4 (Similar Density). Given a frequent dense subgraph g, V_i is the vertex in g. M is another vertex and $M \not\subset V_i$. The similar density between M and g is m/n where n is the number of vertices in g and m is the number of real exist edges between V_i and M, denoted as $dense(M, V_i)$.

The problem of mining frequent dense subgraphs is formulated as follows: given a relation graph dataset, $D=\{G_1,G_2,\cdots,G_n\}$, discovery subgraphs g that satisfy the following two criteria simultaneously: (1) support(g) is higher than a frequent threshold; and (2) extending a vertex M into an existed frequent dense subgraph must be satisfy the similar density threshold. Based on biological

consideration, given a graph dataset, it could not be very useful to produce all of the frequent dense subgraphs. So we find the maximal frequent dense subgraphs.

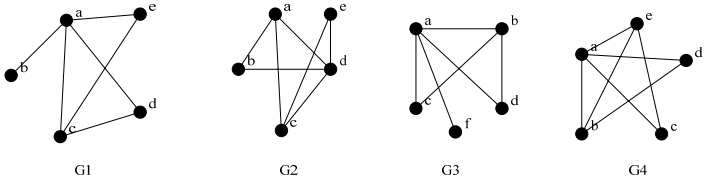

Figure 2. Shown are an example imbalanced networks

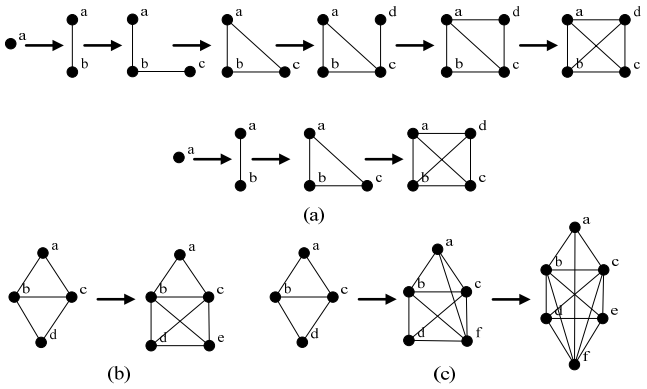

Figure 3(a) Subgraph is obtained by edge-growth and vertex-growth respectively. (b) An example of Lemma 1. (c) Shown example of Lemma3.

III. MINING MAXIMAL FREQUENT DENSE SUBGRAPHS

A. Generating Dense Subgraph

In our approach, the dense subgraph can be obtained by extending a vertex into the existed subgraphs. Comparing to edge-growth method, vertex-growth approach can grow several edges at the same time, as shown in Fig. 3a. Based on the definition of similar density, we make the following definition and lemma on the completeness of generating the dense subgraphs.

Lemma 1. If a vertex *a* can be extended to the existed subgraph *G*, *a* and each vertex in *G* should also satisfy the similar density in the new generated subgraph.

Proof. Based on the definition of similar density, the extended vertex *a* must satisfy the dense threshold. After *a* is extended, *G* must be changed and become larger. Therefore, the original vertex of *G* may not satisfy the dense threshold. So these vertices must be checked whether satisfy or not. If not, *a* can not be extended to *G*.

Lemma 1 guarantees the integrality of generating dense subgraph. For example, in Fig. 3b, based on the dense threshold 0.6, *e* can be extended to subgraph *{a,b,c,d}*. However, if extending *e*, *dense(a, {b,c,d,e})* is 0.5, which can not satisfy the threshold, so *e* should not be extended.

Lemma 2. If a vertex *a* can not be extended to the existed subgraphs *G*, it may be extended to another vertex *Q*, which is the superset of *G*.

Proof. Supposed *dense(a, G)* is *m/n* which is less than the threshold. If *G* can extend other vertices and generate the new larger subgraph *Q*, if *a* is adjacent to the vertex, *dense(a ,Q)* would be *(m+1)/(n+1)*, greater than *m/n*. So there may exist a dense subgraph which can extend *a* successfully.

For example, shown in Fig. 3c, the similar density is 0.5, the current dense subgraph is *{a,b,c,d}*, denoted as *G*. The vertex *e* can not be extended to *G*, while *f* can be extended to *G*, and generate a larger subgraph *{a,b,c,d,f}*, named as *Q*. Now, the vertex *e* can be extended to *Q* and get the largest subgraph *{a,b,c,d,e,f}*.

Traditional cluster algorithms to generate clusters based on the seed vertex, which can be obtained by the vertices distribution. It may be done in one graph. However, generating clusters in multiply graphs, it may not be done easily. Since the distribution of vertices may not be same in each graph. Our MFC can extend any vertex to generate dense subgraphs without using the seed.

Lemma 3. The dense graph can be obtained by extending any vertex of it.

Proof. According to the lemma 2, in the dense graph, a vertex can not be extended by current dense subgraph, it may be extended by another dense vertex. In another word, the vertex is independent from the subgraph. So no matter any subgraph, it can find a subgraph to extend this vertex. Therefore, the first extending vertex can be anyone of graph.

The lemma 3 guarantees the seed vertex is free to be chose. It can increase the efficiency of MFC.

B. Maximal Frequent Dense Subgraph Generation

In this section, we introduce our extending vertex approach to mining maximal frequent dense subgraphs.

There are two approaches for mining maximal frequent subgraphs or patterns: (1) find all the frequent subgraphs, and only output the maximal ones; (2) find a maximal subgraph candidate set and prune the subgraphs that are not maximal. When a new frequent subgraph is discovered, it would check the previous subgraphs to see whether the new one is maximal or not. Therefore, the computing is very time consuming. In our method, MFC exploits the second approach. In order to improve the efficiency, we use several techniques to achieve efficient mining without storing the generated subgraphs.

Definition 5 (Degree). Given a graph G, a is a vertex in G. The degree of a is the number of edges which connect a.

Definition 6 (Unused and used edge). Given a graph G, a and b are adjacent vertices in G. If the edge *{a-b}* is included in a maximal subgraph, this edge is said to be used edge. Or else, it is unused edge.

Lemma 5. If the current subgraph has only one vertex which does not have unused edges, it is unnecessary to extend other vertices to this one.

Proof. The used edge means it is included in a maximal subgraph, if the edges of a vertex are all used, it can not find another maximal subgraph including the existed maximal one.

Lemma 5 can reduce the maximal subgraph check computing. As shown in Fig. 1c, if the subgraph *{a,b,c,d,e,f}* denoted as G, is generated. All the edges of vertex b are included in G, so it is impossible to obtain another subgraph that includes G. Therefore, it is unnecessary to extend b. So does the vertex c, etc.

Lemma 6. If the current extending subgraph has the unused edge, any vertex can be extended to the subgraph, no matter it has unused edges or not.

Proof. Unused edge denotes it is not included in any existed subgraphs, so generating subgraph extended from current subgraph is also not included in any existed ones.

Lemma 6 not only guarantees MFC can generate all the maximal dense subgraphs, but also insures the generated maximal dense subgraphs are same from extending any vertex firstly. We take Fig. 4a for example. Supposed the dense threshold is 0.6, and the dense subgraph *{b,c,d,e,f}* is generated by extending the vertex d, shown in Fig. 4b. a is the current extending vertex and both edges of it are unused edges. So *{a,b,f}* can be generated. Though c, d and e don't have the unused edges, yet *{a,b,f}* is a new subgraph, so c, d and e can be extended to *{a,b,f}*. As shown in Fig. 4c, the three new maximal dense subgraphs are obtained, which is *{a,b,f,c}*, *{a,b,f,d}* and *{a,b,f,e}*, respectively. If extending from a firstly, the same four maximal dense subgraphs can be obtained. The order of generating subgraphs is opposite to extending d in first.

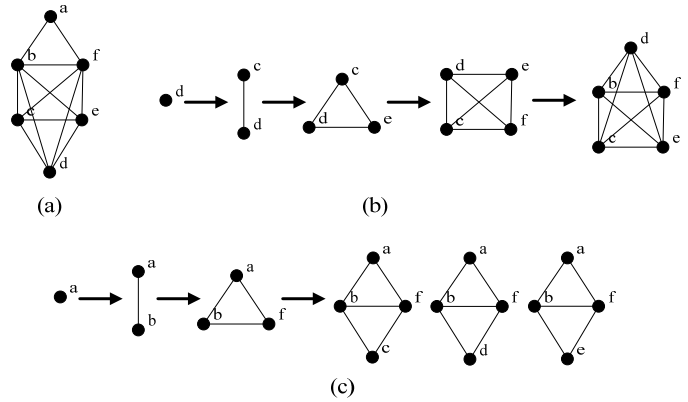

Figure. 4(a) An example dense graph. (b) The process of generating subgraph by extending vertex d.
(c) Three dense subgraphs are obtained by extending vertex a.

However, Lemma 6 can not prevent to produce redundant subgraphs. For example, as shown in Fig. 1c, the current extending subgraph is *{a,b,c,d}*, which can extend vertices e and f. Using depth-first method, *{a,b,c,d,e}* and *{a,b,c,d,e,f}* can be generated. Based on Lemma 6, the subgraph *{a,b,c,d,f}* may be obtained also. Since the vertex f would consider *{a,b,c,d}* be a new subgraph. In the same way, *{a,b,c,d,e,f}* can be obtained again. Therefore, Lemma 6 can not avoid above situation.

Definition 7 (one family subgraph). Given a graph G, a is the first extending vertex. It is said all the dense subgraphs direct or indirect extended from the first vertex a are in one family subgraph.

Definition 8 (family flag). Each edge of family subgraphs would be set a family flag, which is used to differ whether it is used by this family.

Definition 9 (used family edge and unused family edge). In one family subgraph, if an edge is extended by any subgraph in the family, it is denoted as the used family edge, otherwise, as the unused family edge.

For example, a is the first extending vertex, the subgraphs *{a,b,c}*, *{a,b,c,d}* and *{a,c,d}* are in one family subgraph. The family flag of them is '*a*', all the edges are labeled as used family edges.

Lemma 7. Producing dense subgraph in one family, if a vertex does not have unused family edge, it should not be extended by other subgraphs in the family subgraph.

Proof. A vertex not having the unused family edges means it is extended by other dense subgraphs in its family. Since the process of MFC is depth-first, so if extending this vertex again, it can not be obtain a new larger subgraph.

LEMMA 7 CAN ESCAPE OF PRODUCING THE REDUNDANT SUBGRAPHS, WHICH IS SHOWN IN ABOVE EXAMPLE. WHEN *{A,B,C,D,E,F}* IS GENERATED, *F* DOES NOT HAVE THE UNUSED EDGES IN THE FAMILY. AND *{A,B,C,D,E,F}* IS IN THE SAME FAMILY SUBGRAPH AS *{A,B,C,D,E,F}*. THEREFORE, IT IS UNNECESSARY TO EXTEND F TO *{A,B,C,D}*.

C. Pruning Techniques

Based on the above lemmas, the MFC exploits several pruning techniques to achieve efficient mining.

Pruning 1. Given the similar density is *a*, the minimum vertex number of dense subgraph is *m*. The vertex whose degree is not larger than *(m/a-1)*, should be cut from original graphs.

This pruning is based on an observation of the structure of dense graph. Given a vertex *v*, the degree of *v* is *d*. From the last section, if a vertex can be extended, the similar density between this vertex and the current extending subgraph must satisfy the threshold *a*. And the number of subgraph must larger than *m*. It can be shown as follow:

*(d+1)*a>=m, d=m/a-1.*

Using this pruning can reduce the size of original graph and increase the efficiency, which is shown in the next section.

Pruning 2. If the current subgraph has only one vertex, which does not have unused edges, it need not generate the larger dense subgraph from this vertex.

Pruning 3. If a vertex does not have unused family edge in one family, it should not be extended by other subgraphs in the family subgraph.

D. Implementation and Example

Using the support and similar density can ensure the produced subgraphs frequent and dense. The algorithm of the MFC is outlined in Algorithm 1 and illustrated as follow:

Algorithm 1: MFC.
Input: A graph dataset: *G*, the minimum support threshold: *min_sup*, and the minimum similar density threshold: *min_dense*, the minimum number of subgraph: *min_num*, the current dense subgraph: *subgraph*, all the maximal dense subgraphs: *Vertexsets.*
Output: The complete set of maximal frequent dense subgraphs Vertexsets.
Initialization:
Vertexsets=∅, *vertexset*=∅, *flag*=true;
Method: Call MFC(*G, min_support, min_density, vertexset, vertexset, Vertexsets*).

(1) **if** *Vertexsets*=∅ **then** scan the graph dataset *G*, delete the non-frequent edges and vertices, **and** *Pruning 1*;
(2) Finding frequent and dense vertexset v_i in *G*;
(3) **if** v_i=∅ and the number of vertexset is larger than *min_num* **then output**(*vertexset*);
for each vertex *v* in v_i, **do**
(4) **if** *v* satisfies *Pruning 2* or *Pruning 3* **then** *v*=*v*->next;
else *vertexset*=*vertexset*+*v*; *flag*=false;
(5) change the information of vertices and edges in each original graph;
(6) Call MFC(*G, min_sup, min_dense, min_num, vertexset, Vertexsets*);
vertexset=*vertexset*-*v*;
(7) **if** flag=true and the number of *vertexset* is larger than *min_num* **then output**(*vertexset*);

Algorithm 1 illustrates the framework of our frequent dense subgraph mining approach. *Pruning 1* can reduce the size of original graphs, which inceases the efficiency of time and space (*step* 1). In stead of extending edges to grow pattern, we adopts the vertex-based growth method. The value is, when producing a new pattern, we can judge whether it is dense or not (*step* 2). If not, drop it and produce patterns using another path. In each iteration, MFC extends a newly discovered frequent dense graph as much as possible until it finds the largest supergraph(*step* 3), or it can not generate larger subgraph (*step* 7).

Using vertex-growth method may get an interesting question, which is the same vertexsets may contain different graphs (*step* 2). E.g. all the graphs in Fig. 1(a) has the same vertexsets, however, they are not same. In our algorithm, considering one vertexset may contain more different graphs, each of them is independent. If they satisfy the thresholds, we will output all them. While using this method may pose another problem which may generate many similar subgraphs that differ in only a few vertices. Biologically speaking, one protein may have many function, different protein-sets may have different function. One protein may take part in one module to finish a process or function, it may also be joined another module in another time. So we consider mining these similar modules is interesting. The other reason may be the incomplete and inaccurate character of PPI, edges and vertices in original graph may not be all existed in real. A little differ in similar subgraphs may induce to different function in huge. Of course, if two graphs had the same vertexset, they could be considered the same one. Using this method can mine the approximate patterns, which is our recent work.

MFC generates maximal frequent dense subgraphs using depth-first principle (*step* 6). *Pruning 2* and *Pruning 3* can reduce the times of iteration (*step* 4), only the maximal subgraphs can be output. If a vertex can be extended to the current subgraph, related information should be changed in original graphs (*step* 5), such as used edge, used family edge, etc.

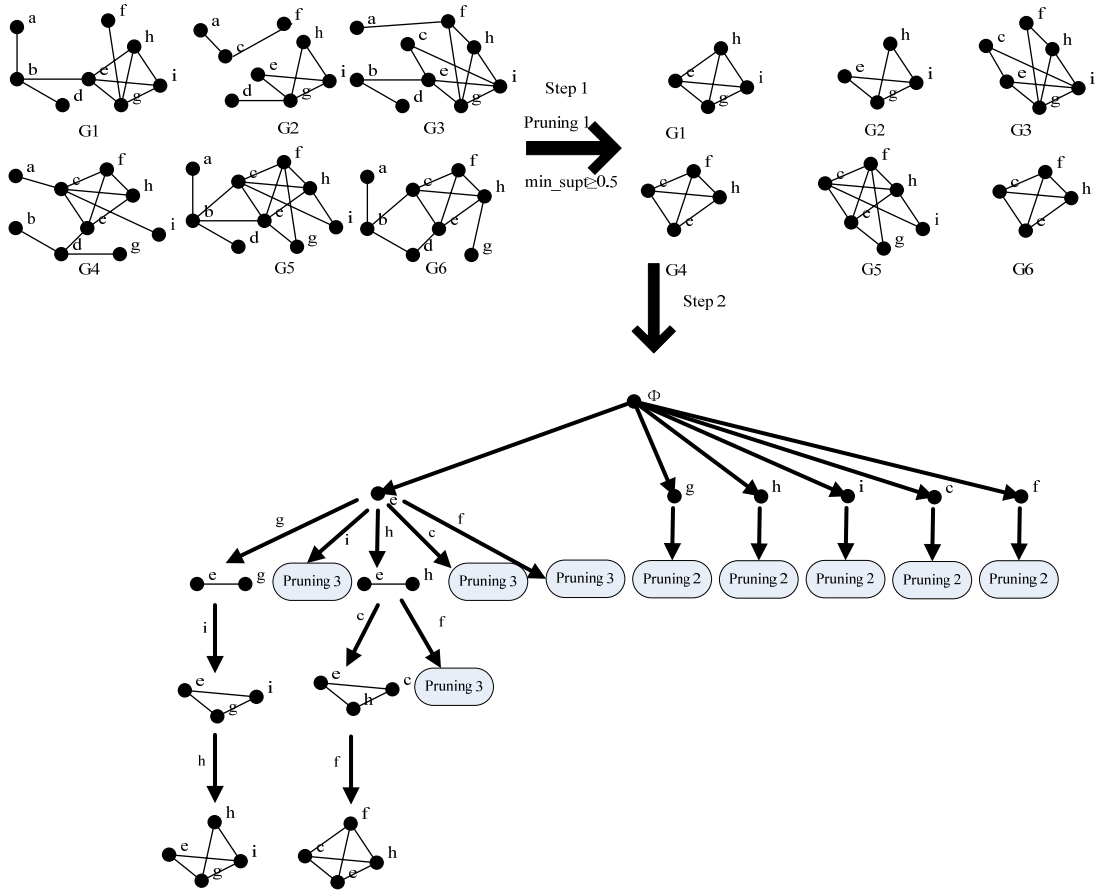

Figure 5. MFC: discovery frequent dense subgraphs across multiple graphs.

Then, we will illustrate the algorithm using the following example which can be found in Fig. 5. We only use absolute support to show how MFC works, using relative support is the similar way. For the graph database *G* in Fig. 5 with *min_sup=0.5*, *min_dense=0.6* and *min_num=3*.

1. Delete the edges and vertices in each original graph which are not frequent and not satisfy the *Pruning 1*. After this step, it can reduce the size of original graphs.

2. Find length-1 subgraph. Scan *G* once to find all the frequent vertices. Each of these frequent vertices is a length-1 subgraph. They are <*e*>*:1.0*, <*g*>*:0.67*, <*h*>*:0.83*, <*i*>*:0.67*, <*c*>*:0.5* and <*f*>*:0.67*, where the notation "<*vertex*>*:decimal*" represents the vertex and its absolute support. The list order does not influence the result.

3. Divide search space. The complete subgraphs can be partitioned into the following six family subgraphs: 1) the subgraphs with ancestral <*e*>, 2) the ones with ancestral <*g*>,…, and 6) the ones with ancestral <*f*>.

4. Find maximal frequent dense subgraphs. The subgraphs can be mined in each family subgraph recursively.

a. Find subgraphs with ancestral <*e*>. The vertex <*e*> is extended firstly. The candidate vertex is <*g*>, <*i*>, <*h*>, <*c*> and <*f*>, respectively. 1) Then <*eg*> would be extended. And the edge <*e-g*> is labeled as used in G and denoted as 'e' for family flag, which means it is used in this family subgraph. There is only one candidate vertex <*i*> of <*eg*> and generates subgraph <*egi*>. The related edges are labeled. In the same way, <*egih*> can be generated. No candidate is found. Output the subgraph <*egih*> and its edges. 2) <*ei*> would not be generated. Since edge <*e-i*> is labeled as used family edge, based on *Pruning 3*, it should be cut. 3) <*c*> and <*f*> can be extended to <*eh*>. The label of related edges would be changed. <*ehc*> can be obtained by extending <*eh*>. Then we get <*ehcf*> which is the maximal subgraph and be output. Based on *Pruning 3*, <*f*> should not be extended to <*eh*>. 4) According to *Pruning 3*, <*c*> and <*f*> are pruned from <*e*>.

b. Find subgraphs with ancestral <*g*>, <*h*>, <*i*>, <*c*> and <*f*>. According to *Pruning 2*, all the vertices should not extending other vertices, since it can't find larger subgraph than existing ones.

E. Analysis

MFC adopts the vertex-growth method, in which one iteration may generate more frequent edges. In addition, it exploits several pruning techniques to avoid storing the generated maximal frequent dense subgraphs and prune unmaximal subgraphs in time. In comparison with previous frequent graph mining algorithms, our algorithm may show significant advantage in memory and computing efficiency.

Mining overlapping dense subgraphs is one of the problems in graph mining. Using MFC, we can solve it easily. For example, when dealing with the graph in Fig. 1b, we can get two subgraphs *{a,b,c,d,e,f}* and *{e,d,g,h}* by extending vertex *a* and *e* respectively. The process is shown in Fig. 6. And for the graph shown in Fig. 7, which can be found in [23], the two dense subgraphs *g1* and *g2* might not be separated accurately by traditional clustering algorithm. However, it can be done easily by our algorithm.

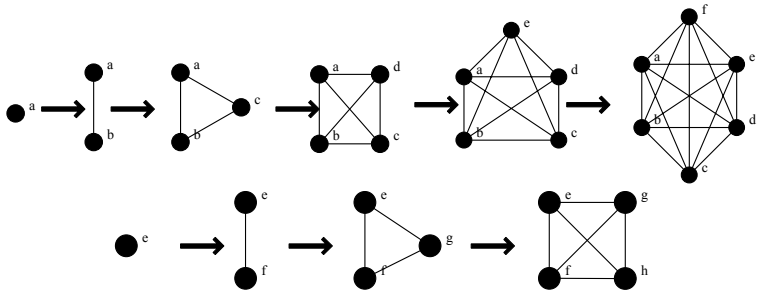

Figure 6. MFC can identify the overlapping subgraphs.

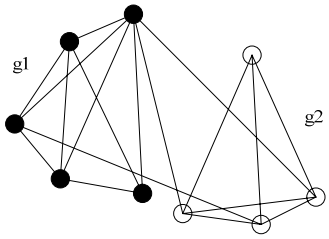

Figure 7. It is observed that two subgraphs *g1* and *g2* might not be separated easily.

IV. MINING MAXIMAL FREQUENT DENSE SUBGRAPHS

A. Data Source and Analysis

In this study, the human PPI networks are used as a testing system for MFC. We integrate four human PPI datasets, which is *DIP* [24], *REACTOME* [25], *HOMOMINT* [26] and *OPHID* [27], respectively. Each PPI dataset is modeled as a relational graph where each node is a unique protein and if an edge exists between two proteins, it is only one. *DIP* and *REACTOME* are literature-based interaction maps. *HOMOMINT* and *OPHID* are orthology-based interaction maps. As shown in Fig. 8, the difference between each dataset is huge.

We apply MFC to discovery frequent dense patterns across the above two class networks respectively: Literature and Orthology, each of them has two PPI networks. The support is 0.75, the minimal number of module is four and the similar density is 0.85. To quantify the comparison, we assess the pattern quality by determining the percentage of functionally homogeneous patterns among all identified patterns. We use the Gene Ontology (GO) annotation to assess our results. If the ratio of cluster members having the same known annotations which belong to a specific GO functional category is greater than the threshold, the subgraph is claimed as homogeneous one. As shown in Fig. 9, due to the imbalanced datasets quantity, Orthology-based PPI can be discovered more annotated modules. From the above figures, we can see different kind PPI network contains different information. So the imbalance character is more important in the analysis of PPI.

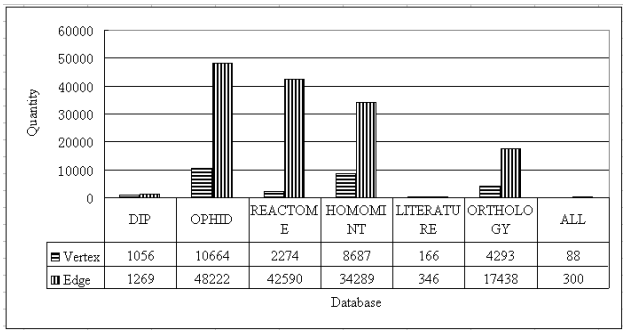

Figure 8.Number of vertices and edges of each dataset.

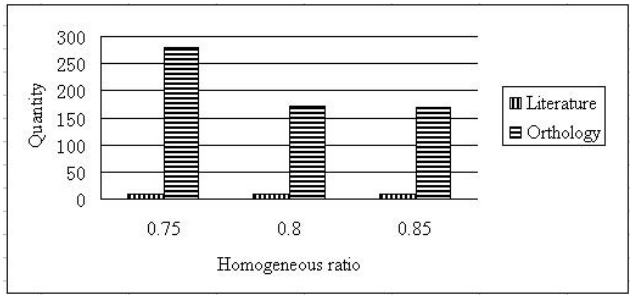

Figure 9. Number of proteins in homogeneous modules assessed by GO.

B. Functional Module Discovery by Relative Support VS Absolute Support

We apply MFC to discover frequent dense subgraphs in the four PPI networks. The similar density is set to be 0.85. In this part, we only focus on the frequent dense subgraphs with at least four vertices. The support is 0.75. From Fig. 10, it can be seen using relative support can discover more dense modules in different support ratio threshold, which is assessed by GO. Relative support-based method is effective to find more annotated modules. Of course, before using relative support to mine, we use the absolute support which is 0.5 to reduce the noise vertices and edges.

To assess the prediction accuracy of our algorithm, we also employe the 'leave-one-out' approach by masking a known protein to be unknown and assign its function based on the remaining known protein in the pattern. We only assess the GO annotated modules. If the function of pattern is the same as the real function of the mask protein, it is considered a prediction to be correct. As shown in Fig. 11, the support is 0.75, more accurate function of proteins can be predicted using relative support.

By applying this approach to the previous version of GO, we make a functional prediction for some proteins. The prediction can be confirmed by the recent version. By applying our method to previous GO database(2005-4-18), the protein O15111 can be predicted to be involved in GO:0005634, because all of the remaining four proteins in the same subgraph participate in that cellular component, which is shown in Fig. 12.

C. Comparison with Existed Approach

In this section, we are going to compare a summary graph-based approach-CODENSE and a dense subgraph mining algorithm-MODES to demonstrate the effectiveness of our method. CODENSE is a novel algorithm which can efficiently mine frequent coherent dense subgraphs across large number of massive graphs. It uses MODES to cluster the summary graph. The detail can be found in [22].

Firstly, MFC and CODENSE will be compared. The support is 0.75 (MFC uses the absolute support), the density of subgraph is 0.85, the number of subgraph is at least 3. We also use GO and 'leave-one-out' methods to assess the results. As shown in Fig. 13, MFC can find more annotated modules, which is more efficiently than CODENSE. From the Fig. 14, assessed by 'leave-one-out', more accurate function of proteins can be predicated by using our algorithm. Then we test the performance of finding dense subgraphs between MFC and MODES in single dataset. The parameters are the same as above. As shown in Fig. 15, MFC is more efficiently than MODES in running time in all datasets but REACTOME, since REACTOME has more dense subgraphs than other datasets. Due to the stronger power of finding dense subgraphs, MFC may have less efficiency than MODES in this dataset. Fig. 16 shows our algorithm can find more modules than MODES.

D. Scalability Study

We also evaluate MFC scalability using the enlarged dataset which is replicated by 5, 10, 15 and 20 times. Shown in Fig. 17, it is evident that MFC shows the linear scalability in runtime against the number of replicated datasets.

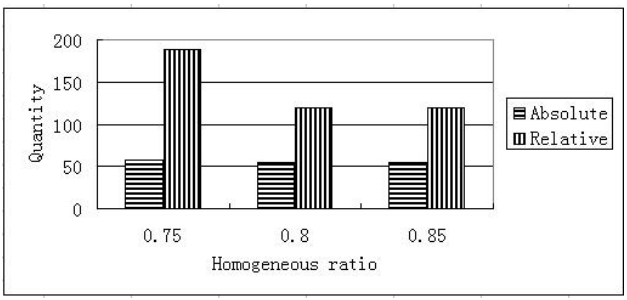

Figure 10. Number of proteins in homogeneous modules assessed by GO, support is 0.75.

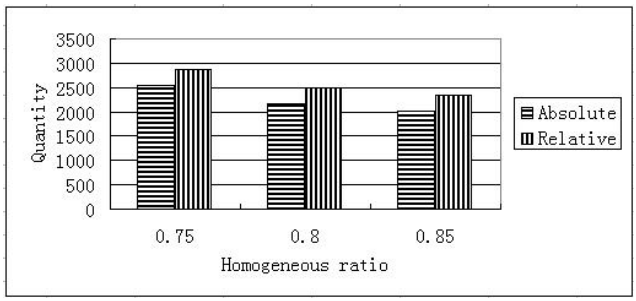

Figure 11. The number of protein function assessed by 'leave-one-out'.

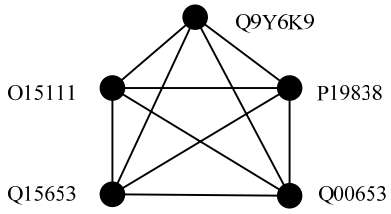

Figure 12. All four proteins except O15111 are known to be involved in cellular component.

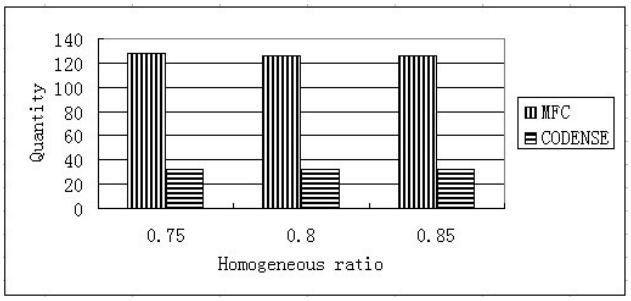

Figure 13. Number of proteins in homogeneous modules assessed by GO.

Figure 14. The number of protein function assessed by 'leave-one-out'

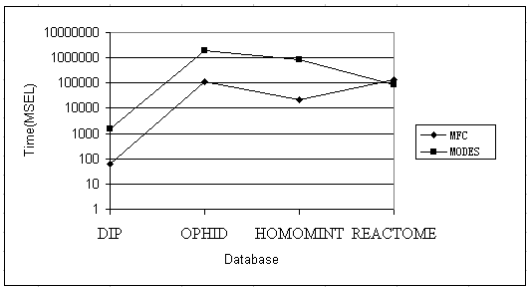

Figure 15. The running time in each dataset.

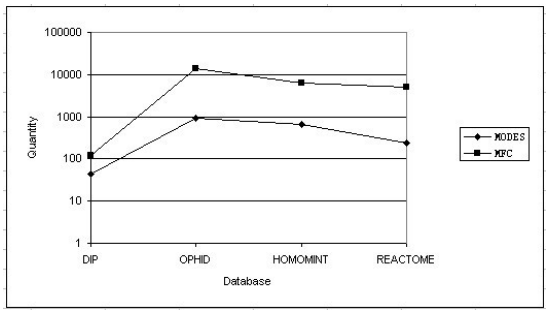

Figure 16. Number of dense modules.

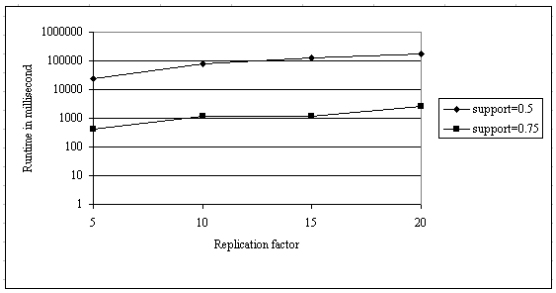

Figure 17. Scalability in each replicated dataset.

V. CONCLUSIONS

We propose an algorithm, MFC, to efficiently mine frequent dense subgraphs across massive imbalanced protein-protein interaction networks. Instead of using summary graph, MFC mines frequent patterns by extending vertices, which can reduce the generation of false patterns as well as can mine overlapping graph clusters. We introduce relative support to mine frequent patterns in consideration of imbalance of PPI datasets. An extensive performance study using real datasets illustrated that the algorithm MFC is efficient and effective. As future work, we will plan to mine frequent dense subgraphs by integrating the PPI datasets and gene expression datasets.

ACKNOWLEDGMENT

The work is supported by the National Natural Science Foundation of China under Grant No.60703105. It is also partly supported by the Natural Science Foundation of Shaanxi Province under Grant No.2007F27. Preliminary works reported in this paper was presented at the Second International Workshop on Intelligent System and Applications (ISA 2010)[28] and the Third International Conference on Bioinformatics and Biomedical Engineering (iCBBE 2009)[29].

REFERENCES

[1] Waterman, M.S., Galas,D.J. and Arratia,R: Pattern recognition in several sequences: consensus and alignment. Bull. Math. Biol., 46, 515-527, 1984.

[2] Wu,T.D. and Brutlag,D.L: Identification of protein motifs using conserved amino acid properties and partitioning techniques. In Proceedings of the 3rd International Conference on Intelligent Systems for Molecular Biology. AAAI Press, Menlo Park, CA, pp. 402-410, 1995.

[3] Neville-Manning,C.G., Sethi,K.S., Wu,D. and Brutlag,D.L: Enumerating and ranking discrete motifs. In Proceedings of Intelligent Systems for Molecular Biology. AAAI Press, Menlo Park, CA, pp. 202-209, 1997.

[4] Smith,T.F. and Waterman,M.S: Identification of common molecular subsequences. J. Mol. Biol., 147, 195-197, 1981.

[5] Smith,H.O., Annau,T.M. and Chandrasegaran,S: Finding sequence motifs in groups of functionally related proteins. Proc. Natl Acad. Sci. USA, 87, pp. 826-830, 1990.

[6] Suyama,M., Nishioka,T. and Jun'ichi,O: Searching for common sequence patterns among distantly related proteins. Protein Eng., 8, 1075-1080, 1995.

[7] Wang K, Hu Y, Hu Yu J. Scalable Sequential Pattern Mining for Biological Sequences. Proceedings of the 13th ACM conference on Information and knowledge management, USA. 2004; p. 178–87.

[8] Aebersold R, Mann M: Mass spectrometry-based proteomics. Nature 2003,422: 198–207.

[9] Fields S: High-throughput two-hybrid analysis. The promise and the peril. FEBS J 2005,272: 5391–5399.

[10] Yan,X., Zhou,X. and Han,J: Mining closed relational graphs with connectivity Constraints. Proceedings of the International onference on Data Engineering, Boston, MA, March 30–April , 2005. IEEE Computer Society.

[11] Chua HN, Sung WK,Wong L : Exploiting indirect neighbours and topological weight to predict protein function from protein–protein interactions. Bioinformatics2006, 22: 1623–1630.

[12] Nabieva E, Jim K, Agarwal A, Chazelle B, Singh M : Whole proteome prediction of protein function via graph-theoretic analysis of interaction maps. Bioinformatics 21 , 2005, (Suppl 1): i302–i310.

[13] Deng M, Zhang K, Mehta S, Chen T, Sun F: Prediction of protein function using protein–protein interaction data. J Comput Biol , 2003 10:947–960.

[14] Sharan R, Ideker T, Kelley B, Shamir R, Karp RM: Identification of protein complexes by comparative analysis of yeast and bacterial protein interaction data. J Comput Biol 2005,12: 835–846.

[15] Wu DD, Hu X: An efficient approach to detect a protein community from a seed. 2005 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB 2005). La Jolla, CA, USA: IEEE pp. 135–141.
[16] Tsuda K, Shin H, Scholkopf B: Fast protein classification with multiple networks. Bioinformatics 21,2005 (Suppl 2): ii59–ii65.
[17] Spirin V, Mirny LA : Protein complexes and functional modules in molecular networks. Proc Natl Acad Sci USA 2003, 100: 12123–12128.
[18] Przulj N, Wigle DA, Jurisica I: Functional topology in a network of protein interactions. Bioinformatics 2004, 20: 340–348.
[19] King AD, Przulj N, Jurisica I: Protein complex prediction via cost-based clustering. Bioinformatics 2004, 20: 3013–3020.
[20] Krogan NJ, Cagney G, Yu H, Zhong G, Guo X, Ignatchenko A: Global landscape of protein complexes in the yeast Saccharomyces cerevisiae. Nature 2006, 440: 637–643.
[21] Titz B., Schlesner M. and Uetz P: What do we learn from high-throughput protein interaction data? *Expert Rev.Proteomics*, 1(1), 111–121.
[22] Hu,H. et al: Mining coherent dense subgraphs across massive biological networks for functional discovery. Bioinformatics, 21 (Suppl. 1),2005, i213–i221.
[23] Yan et al: A graph-based approach to systematically reconstruct human transcriptional regulatory modules. Bioinformatics, 23, 2007, i577–i586.
[24] Xenarios I, Salwinski L, Duan XJ, Higney P, Kim S, Eisenberg D (2002) DIP: The Database of Interacting Proteins. A research tool for studying cellular networks of protein interactions. *NAR* **30**:303-5.
[25] Joshi-Tope, G.M. Gillespie I. Vastrik P. D'Eustachio E. Schmidt B. de Bono, et al., Reactome: a knowledgebase of biological pathways. Nucleic Acids Res, 2005. 33(Database issue): p. D428-32.
[26] Persico,M., Ceol,A., Gavrila,C., Hoffmann,R., Florio,A. and Cesareni,G. (2005) HomoMINT: an inferred human network based on orthology mapping of protein interactions discovered in model organisms. BMC Bioinformatics, 6, S21.
[27] Brown KR, Jurisica I: Online Predicted Human Interaction Database. Bioinformatics 2005, 21:2076-2082.
[28] M.Wang, X.Q.Shang, et al. Mining maximal frequent dense subgraphs without candidate maintenance in PPI networks. The proceedings of ISA 2010, Volume 1, pp:158-161, 2010.
[29] M.Wang, X.Q.Shang, D.Xie, et al. Mining frequent dense subgraphs based on extending vertices from unbalanced PPI networks. The proceedings of iCBBE 2009, 978-1-4244-2902-8, June 11, 2009, Beijing, China.

Miao Wang is a doctoral student at The School of Computer Science and Engineering at the Northwestern Polytechnical University, Xi'an, China. He completed his Master Degree from University of Northwestern Polytechnique in 2008. Since 2006 he has been researching in data mining and bioinformatics.

Xuequn Shang is a association professor at The School of Computer Science and Engineering at the Northwestern Polytechnical University, Xi'an, China. She completed her PhD degree from University of Magdeburg, Magdeburg, Germany. Since 2001 she has been researching in data mining, database technology and bioinformatics.

Zhanhuai Li is a professor at The School of Computer Science and Engineering at the Northwestern Polytechnical University, Xi'an, China. His research interest is database technology, software engineering and data mining.