Improving Functional Modularity in Protein-Protein Interactions Graphs using Hub-Induced Subgraphs

Duygu Ucar * Sitaram Asur* Umit Catalyurek † Srinivasan Parthasarathy*†‡

Abstract

Protein-protein interaction (PPI) graphs are believed to be fundamental sources of information related to functional behavior of proteins. Hypothetically, dense components extracted from PPI graphs correspond to functional modules of proteins. The main problem with analyzing PPI graphs is their scale-free topology which makes it hard to isolate dense components effectively. In this paper, we present a refinement method based on induced neighborhoods in order to improve the functional modularity of a PPI graph. The resultant graph is less scale-free and can be partitioned effectively into dense components. A detailed comparison of these dense components with the ones obtained from the original graph show three major benefits: i) Isolation of additional functional groupings; ii) Enhancement of existing functional groupings; and iii) Fuzzy-clustering of certain multi-functional proteins allowing them to belong to multiple clusters.

1 Introduction

Protein-Protein interaction (PPI) graphs are accumulations of experimentally determined interactions between proteins. These graphs are important sources for the discovery of functional groups of proteins. The presence of biologically relevant functional modules in PPI graphs has been theorized by many researchers [7, 16, 29]. A functional module can be defined as a group of proteins that take part in the same biological function. The task of extracting these modules for the purposes of functional prediction and identification is an active research area in functional genomics.

The primary property of the PPI graph that is detrimental to traditional graph partitioning is its scale-free topology [26]. This means that the degree distribution of the graph follows a power law function, as $P(k) \sim k^{\alpha}$ where $P(k)$ is the number of nodes with degree k and $\alpha < 0$. It implies that most proteins in the graph participate in a small number of interactions while a few proteins, known as hubs, are involved in a large number of interactions. The topology typically consists of a giant central core containing a significant amount of proteins and their interactions. The rest of the proteins are either completely disconnected or part of small disconnected groups. The functional modules are hidden inside the giant central core. The scale-free topology, coupled with the tendency of the hubs to interact with a high fraction of proteins, makes isolation of modules all but impossible [23]. Hence, traditional graph partitioning algorithms fail to effectively isolate functional modules within these graphs.

Another challenge in partitioning protein interaction graphs is the need to assign proteins to different groups based on their different functions in the cell. The hub proteins are likely to be essential for the organism and typically have multiple functions. Traditional partitioning algorithms fail to find these essential proteins and assign them to multiple groups. Recently, Karypis *et al* [2]; presented new multi-level graph partitioning algorithms to address the difficulty of partitioning scale-free graphs. Although the proposed algorithms result in better partitions, they still do not have the ability to distribute some proteins into several partitions.

In order to address these issues, a refinement of the PPI graph is suggested in this article. The main point motivating this work is the multi-functional nature of essential proteins (hubs) [17]. Our goal is to target these multi-faceted hubs - to isolate, for each of their functions, the groups of proteins in the neighborhood of each hub that share the same function. Our approach is based on two stages. In the first stage, we refine the PPI graph, reducing the scale-free property, perform duplication of hub nodes emphasizing their neighborhoods. We present two strategies based on graph cores [24] and Edge Betweenness [21] for this purpose. In the second stage, we partition the refined graph using suitable graph partitioning algorithms and obtain dense components. The suggested technique, thus aims to isolate dense components with high degree of overlap

*Department of Computer Science and Engineering, The Ohio State University

†Department of Biomedical Informatics, The Ohio State University

‡To whom correspondence should be addressed. srini@cse.ohio-state.edu

with known functional modules. Other researchers have studied elimination of hubs from the scale-free graphs, but found in most cases, that this disconnects the graph and breaks down the modules as well [3, 10]. To the best of our knowledge, we are the first to suggest hub duplications to improve modular decomposition of PPI graphs. The refinement process is also designed to perform fuzzy clustering of the hub proteins, which is in accordance with the multi-functional nature of hubs.

This refinement technique is tested on the PPI graph of budding yeast (Saccharomyces Cerevisiae) obtained from the DIP (Database of Interacting Proteins) database. We find that the partitions we obtain after refinement match well with known functional modules. Our results also showed that the refinement technique allows multi-functional proteins to be assigned to multiple partitions. Each of these partitions include several proteins sharing a certain function with the multi-functional protein.

In order to quantify the quality of partitions in terms of their functional similarity we employ the Gene Ontology (GO) consortium database. This database provides structured vocabularies (i.e., ontologies) to annotate genes in terms of their associations in biological processes, molecular functions and cellular components. Biological process and molecular function are recognized to be more relevant to functional modules of proteins and are hence, considered for the evaluation purposes.

2 Graph Refinement

2.1 Evolutionary Implications: Recently, several groups have suggested mathematical models to explain the evolutionary growth of Protein-Protein interactions graphs [5, 9, 27]. According to these, gene duplication and preferential attachment are the main causes for the scale-free topology of interaction graphs. Each gene duplication event leads to a new protein that interacts with a subset of neighbors of the ancestor gene. Vazquez *et al* [27] propose a duplication-divergence model for evolution of PPI graphs. In the duplication stage, a new node is created and linked to all neighbors of the parent node with probability p. In the divergence stage, some of these links are removed with probability q. It is important to note that these new nodes may be connected to other nodes but with insignificant probabilities. According to this model, the probability that the degree of a node increases by one is given by

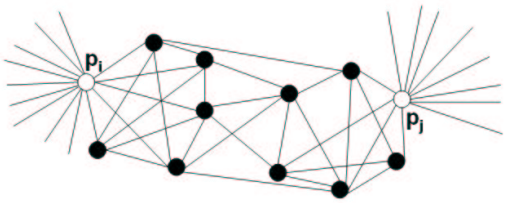
$$w_{kN} \sim (1-q)\frac{k}{N} \tag{2.1}$$

where k is the degree of the node and N is the total number of nodes in the graph. This implies that there is a linear relation between a node's degree and probability of a new node attaching to that node. Since we know that hubs have very high degrees, new proteins that are added to the graph are more likely to have interactions with hubs rather than other nodes. If a hub belongs to a functional module, due to preferential attachment, most of the other proteins of that module will prefer to connect to the hub rather than a node with the same function but less degree. This suggests that proteins with the same function are connected within themselves and they are also individually connected to at least one hub. Hence, we believe that it is important to consider neighborhoods of hubs, to isolate functional modules.

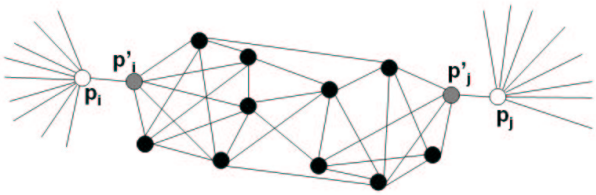
2.2 Hub-induced Subgraphs: As stated before, hubs typically tend to be essential proteins [17], having several important functions inside the cell. Therefore, hubs can be connected to several functional modules at the same time. According to the notion of preferential attachment, functional modules are likely to appear in direct neighborhood of one or more hubs, at least partially. Since, we know that hubs are connected to large number of proteins, most of these interactions do not imply a functional similarity. In this work we are aiming to identify the neighbors of hubs that share functionalities with the hub protein. Hence, our goal is to identify all dense components that lie within a hub's neighborhood. Once such components are identified, the neighboring hub is duplicated and all its interactions with the members of the group are reassigned to the duplicate. In addition, all the duplicates will be connected to the original hub to preserve the interactions of the proteins belonging to the isolated dense component with the other proteins in the neighborhood. Note that we are not eliminating any of the interactions. We are merely reassigning interactions between the proteins belonging to the dense components and the hub to the duplicate.

Notice that, if the proteins of a functional module are divided across neighborhoods of several hubs, each of those hubs will be duplicated once and will be included in the functional module. As a result, the functional module will be isolated from the unrelated neighbors of the hubs and appear as a more tightly knit group. An example can be seen in Figure 1.

Here, we investigate the effect of breaking up a hub into several new nodes based on each dense component in the hub's neighborhood. In order to identify these dense components, we come up with the definition of a hub-induced subgraph. Let $G = (V, E)$ be a graph. $G'(V', E')$ is a vertex-induced subgraph of G if $V' \subseteq V$ and E' is all the edges of G having both endpoints in V'. An example graph and it's vertex-induced subgraph is shown in Figure 2.We define a hub-



(a) Example graph



(b) Graph after duplications

Figure 1: **Functional modules are isolated from the hubs' unrelated neighbors by the use of hub duplications. In figure (b), proteins p_i and p_j are duplicated and the duplicates, p_i' and p_j' are connected to dense components as well as the original proteins.**

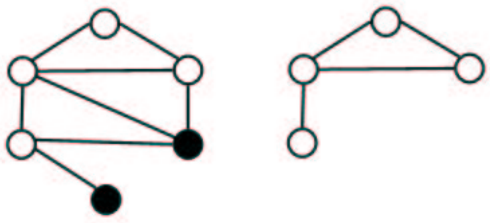
induced subgraph of G as a graph $G''(V'', E'')$, where V'' corresponds to a hub's adjacency list. Thus, for every hub of the graph there exists a corresponding hub-induced subgraph induced by that hub's adjacency list.

Subgraphs induced by the neighbors of each hub are isolated for further analysis. Our observations indicate that these subgraphs have very different topologies. The Clustering coefficient [28] can be used to express the topological dissimilarity of hub-induced subgraphs. It is a measure that represents the interconnectivity of a vertex's neighbors. The clustering coefficient of a vertex v with degree k_v can be defined as follows:

$$CC(v) = \frac{2n_v}{k_v(k_v - 1)} \tag{2.2}$$

where n_v denotes the number of triangles that go through node v. The clustering coefficient of a graph is the mean clustering coefficient over all vertices in it. Clustering coefficient values lie between 0 and 1. Tightly knit groups are associated with high clustering coefficients.

Some of these hub-induced subgraphs are highly connected, including one or more dense subcomponents, whereas the others are almost entirely disconnected.



(a) Graph G (b) Vertex-induced subgraph G'

Figure 2: **Subgraph in (b) corresponds to the vertex-induced subgraph of graph G induced by the white vertices.**

These observations are in accordance with the hub distinction made by Han *et al* [12]. They observed two types of hubs- *party* hubs and *date* hubs. Neighbors of *date* hubs are significantly more diverse in spatial distribution than neighbors of *party* hubs. Their study indicated that the removal of *party* hubs does not affect connectivity and resembles failures, whereas attacks directed against *date* hubs account for a vast majority of the effect observed when attacking all hubs. So, hubs with loosely connected induced subgraphs corresponds to *date* hubs which have neighbors all around the graph. For our purposes, *party* hubs are the ones to be duplicated because their neighbors form one or more densely connected subgroups probably sharing a functionality.

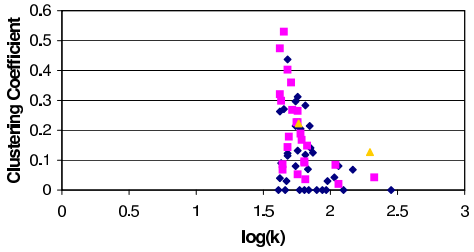


Figure 3: **Log(degree) of 60 hubs vs. clustering coefficients of the corresponding hub-induced subgraphs. Squares correspond to essential proteins whereas diamonds are associated with non-essential proteins in the GENECENSUS database. There are 2 proteins for which we are not able to find essentiality information (shown with triangles).**

In our work, we define nodes with significantly large degrees to be hubs. The DIP dataset consists of 15147 interactions between 4741 proteins. Accordingly, we considered 60 nodes with degrees ranging from 40 to 283 as hubs. Each of these 60 hubs is associated with a hub-induced subgraph. Figure 3 illustrates the log(degree)

of each hub versus the clustering coefficient of the corresponding hub-induced subgraph. Ten of these subgraphs have zero clustering coefficient implying that they completely lack an interacting triplet. We find that all these subgraphs are associated with non-essential proteins according to the GENECENSUS database [19]. These results suggest that hub-induced subgraphs can be used to distinguish essential and non-essential hubs.

2.3 Graph Core Duplication To obtain information about which hubs to duplicate, we use the concept of graph cores [24]. For a given graph, $G(V, E)$, a subgraph G' induced by $V' \subseteq V$ is a core of order k , iff for every $v \in V'$, $d_{G'}(v) \geq k$,where $d_{G'}(v)$ is the degree of vertex v in G'. The core number of a vertex v, is defined as the maximum order of a subgraph that contains v. Thus, the k-core of a graph is a subgraph of the given graph, where each vertex has at least k neighbors in that subgraph. There is a $O(|V|)$-time algorithm to compute the core number of each node in the graph.

Only if a hub-induced subgraph has a k-core, the corresponding hub will be duplicated. The refinement algorithm using k-cores is given below. Here T_k represents the k-core threshold. If a hub-induced subgraph has a core of order $k \geq T_k$, it is identified as a dense component and corresponding hub is duplicated. After the duplication, the interactions between core members and the hub will be reassigned to its duplicate. As implied by the algorithm, graph cores allows only single duplicates for each hub.

Algorithm 1 k-core Duplication(G_i)

```
Let G_i(V_i, E_i) be the hub-induced subgraph for Hub_i
k = maxcore(G_i)
if k ≥ T_k then
  Duplicate Hub_i
end if
```

2.4 Edge Betweenness Duplication Although the graph cores refinement improves the graphs, it is limited to duplicate a hub only once, if any. However, it is quite likely that several dense components associated with different functions are in the direct neighborhood of a single hub. Therefore we decided to employ an algorithm that will be able to identify an optimal number of dense subgroups. The Edge Betweenness graph partitioning algorithm has been used to deduce (partial) functional modules hidden inside the hub-induced subgraphs. The algorithm was first introduced by Newman *et al* [21]. The algorithm is designed to deduce dense subcomponents of a given graph by eliminating edges with high betweenness scores in an iterative manner. More formally, given a graph, $G(V, E)$ and number of partitions k, the algorithm identifies k subgroups such that the intra-group connections are dense, but inter-group connections are sparser. The algorithm stops when the given number of partitions is obtained. Although this algorithm is computationally costly, with $O(E^2V)$ worst-case running time, since the hub-induced subgraphs are small in size, at most 283 nodes, it is still tractable.

For our work we need to detect dense subcomponents inside each hub-induced subgraph. However, as we already discussed, the hub-induced subgraphs have dissimilar topologies. That is why each subgraph has different number of dense subcomponents if there is any. We implemented the algorithm without the 'number of partitions' parameter and included two stopping criteria to determine dense subcomponents; number of nodes and the clustering coefficient of the subgraph.

The pseudocode of the Edge Betweenness refinement algorithm with the newly added stopping conditions is given below. Here, *most-between-edge(G_i)* returns the edge with the highest betweenness score in the G_i subgraph. T_{size} and T_{cc} represent the size and clustering coefficient thresholds, respectively. Whenever a subgraph meets both of these criteria, it is returned as a dense subcomponent. For each dense component, the corresponding hub is duplicated once and its interactions with the dense component's members are reassigned to the duplicate.

Algorithm 2 Edge-Betweenness-Duplication(G_i)

```
Let G_i(V_i, E_i) be the hub-induced subgraph for Hub_i
if size(G_i) < T_size then
  Return
else if CC(G_i) ≥ T_cc then
  Duplicate Hub_i
else
  e = most-between-edge(G_i)
  remove e from G
  G_i ← G_i − e
  if G is partitioned into G_i^1 and G_i^2 then
    Edge-Betweenness-Duplication(G_i^1)
    Edge-Betweenness-Duplication(G_i^2)
  else
    Edge-Betweenness-Duplication(G_i)
  end if
end if
```

3 Graph Partitioning

After the PPI graph is refined using hub-induced subgraphs, the resulting graph is partitioned to separate out the functional groups of the organism. We have used

a single-level spectral partitioning algorithm and two multi-level partitioning algorithms available in Chaco (version 2.0) [15] and METIS [18] graph partitioning tools.

Spectral-based algorithms use eigenvectors of the Laplacian matrix constructed from the graph to determine effective partitions of the graph. These partitions minimize the total weight of the edge cut. Chaco implements a spectral quadrisection algorithm that uses the second and the third smallest eigenvectors respectively to partition the graph. It divides the graph into four partitions at each stage rather than simply bisecting based on one eigenvector (as in spectral bisection). Multidimensional spectral methods have been shown to have several advantages over spectral bisection [13, 14]. Hence, we used the spectral quadrisection algorithm as implemented in Chaco.

In addition to single level spectral partitioning, we considered two multi-level algorithms as well. Multi-level graph partitioning algorithms iteratively coarsen the original graph and obtain an approximate graph. An initial partitioning is performed on this approximate small graph. This initial partitioning is later used to derive a partitioning on the original graph. Chaco uses a spectral bisection method to partition the approximate graph (Multi-level Spectral algorithm). The kMETIS algorithm, on the other hand, obtains a k-way partition of the approximate graph and refines it to construct a k-way partitioning of the original graph.

4 Experiments

In this section, we discuss the evaluation method and experimental results.

4.1 Validation Method To test if the partitions obtained correspond to known functional modules, we need to validate our dense components using known biological associations. We use the Gene Ontology Consortium Online Database [4] to look for biological relations between proteins that are assigned to the same partition. The Gene Ontology (GO) is a controlled vocabulary designed to accumulate the result of all investigations in the area of genomics and biomedicine by providing a large database of known associations containing common terminology that can be used among researchers.

GO provides three vocabularies namely cellular component, molecular function and biological process. The cellular component terms refer to the localization of proteins inside the cell. The molecular function terms refer to shared activities at the molecular level. Lastly, the biological process terms refer to entities at both the cellular and organism levels of granularity. Among these three, biological process and molecular function are known to be the most relevant to functional modules of interaction graphs. Hence, we use them for validation and comparisons. As of May 2005, the GO database contains 7000 genes annotated with 7502 molecular functions and 9706 biological processes.

For a group of proteins, we query the annotations to identify biological processes and molecular functions that are performed by members of that group. We employ the GO-TermFinder tool [1] for this purpose. However, merely counting the proteins that share a biological process or molecular function will be misleading since the underlying distribution of genes among different annotations is not uniform. Hence, we use p-values to signify a group of proteins that share a GO term. P-values are calculated using the Hypergeometric Distribution which is defined as follows.

If there are N proteins in the database with M of them assumed to share a particular property and our partition contains n proteins out of which m have the same annotation. Then using the Hypergeometric Distribution, we get

$$p = \frac{\binom{M}{m}\binom{N-M}{n-m}}{\binom{N}{n}} \tag{4.3}$$

where $\binom{a}{b}$ represents the number of combinations by which b entities can be selected from a entities.

This formula gives the likelihood of grouping the proteins together by chance. If the p-value is small, then it implies that the grouping is not random. Hence, this grouping is more significant biologically than one which has a higher p-value. A cut-off value is used to differentiate significant groups from the insignificant ones. If a group of clusters are associated with a p-value greater than the cut-off, they are considered to be insignificant. We have used the recommended cut-off of 0.05 for all our validations.

As the p-value of a single partition is statistically not representative, we define a p-value score in order to quantify the partitioning results. We defined the p-value score as follows.

$$score = \frac{\sum_{i=1}^{n} min(p_i)}{n} \tag{4.4}$$

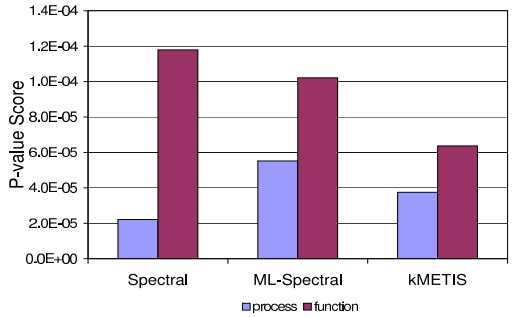
where n denotes the number of partitions with significant p-values (smaller than the cut-off) and $min(p_i)$ denotes the smallest p-value of the partition i.

The Gene Ontology method of validation is not flawless since it is incomplete. However since we are using the same set of annotations to ascertain functional modules before and after the hub duplications, the evaluation will be equally biased before and after

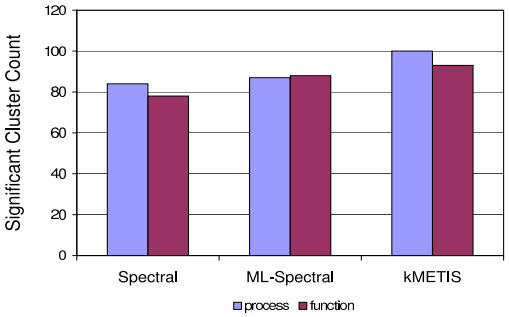
the refinement.

4.2 Experimental Results

4.2.1 Comparison of Partitioning Algorithms We compared the three partitioning algorithms (Spectral, Multi-level Spectral and kMetis) to find the most suitable method to partition the PPI graph. To eliminate any bias, we ran all our algorithms with the same parameters. We picked 128 as the number of partitions since the size of the final partitions are in accordance with sizes of known functional modules.



(a) P-value scores



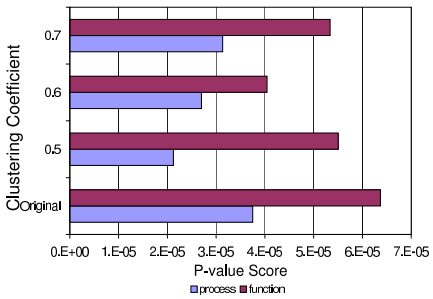
(b) Number of significant partitions

Figure 4: **Comparison of different partitioning algorithms on PPI graph. Figure (a): P-value scores represent the average p-values of the partitions we obtained using different algorithms. Figure (b) indicates the number of partitions whose p-values are smaller than the cut-off(0.05).**

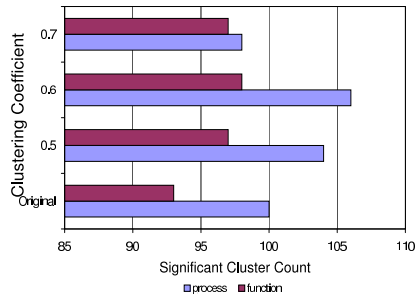
The results obtained using different partitioning algorithms are shown in Figures 4a-4b. As seen from these figures, the kMETIS algorithm outperforms the spectral based algorithms. The number of significant partitions is largest in kMETIS among all three algorithms. In addition it has the smallest p-value score for the molecular function ontology. This can be attributed to the fact that kMETIS is based on k-way partitioning whereas the other two algorithms are founded on recursive two-way and four-way partitions. It is well known that recursive bisection algorithms can do arbitrarily worse than k-way partitioning [25]. Thus, an algorithm which directly obtains k-way partitioning can potentially capture the global topology of the graph better when compared to bisection or quadrisection algorithms. Since, kMETIS performs best overall, we choose it to test the efficiency of our refinement technique.

4.2.2 Edge Betweenness Refinement In this experiment, we test the effectiveness of the Edge Betweenness refinement technique. We ran the algorithm to find all dense subcomponents that are composed of at least 5 nodes. We decided to choose 5 as the minimum size for dense components, since subcomponents of size smaller than 5 are likely to be insignificant. To choose a suitable threshold for the clustering coefficient, we tested various values to obtain the optimum number. There are two things that we should consider while deciding on this parameter. First, we want the resulting subcomponents to be dense enough to correspond to a functional module. So we need to consider sub-components whose clustering coefficient is greater than 0.5(i.e., half of the possible triplets are formed). On the other hand, it is well known that PPI datasets are prone to high false negative rates. Hence, we cannot expect perfect cliques (i.e., all possible triplets are formed) among the hub-induced subgraphs. So, we vary the clustering coefficient parameter between 0.5 and 0.7 and obtain refined graphs for each. These refined graphs as well as the original graph are partitioned by kMETIS separately. The results we obtained are depicted in Figure 5(a). As can be seen from this figure, p-value scores are getting smaller (improving) after the refinement. Although we obtained better p-value scores after refinement, we wanted to test whether the partitioning is, indeed, improving. In Figure 5(b), the number of significant partitions are depicted before and after the refinement. We are able to identify more number of significant partitions after refinement. Although improvement is observed for each clustering coefficient threshold, the improvement is more significant for 0.5 and 0.6. This result might be due to incomplete knowledge about interactions (high false negative rate). Since functional modules do not have interactions between all its members, a very high clustering coefficient threshold is too restrictive to capture all the functional modules. These results suggest that 0.5 and 0.6 are optimal clustering coefficient thresholds. We pick 0.6 for the rest of our experiments.

Although kMETIS outperformed the other algorithms, for the sake of completeness we give results for

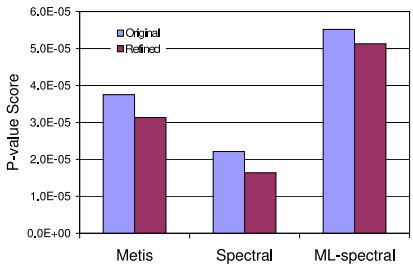


(a) P-value scores

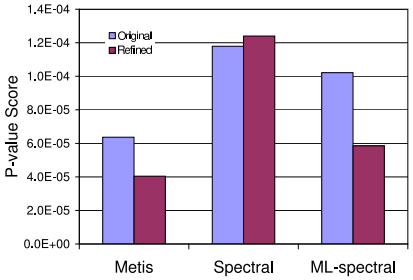


(b) Number of significant partitions

Figure 5: **P-value scores and number of significant clusters before and after Edge Betweenness refinement algorithm is run on DIP dataset. We varied the clustering coefficient threshold and tested the effectiveness of the algorithm in each case. Original refers to the DIP dataset before the refinement. 0.5 ,0.6 and 0.7 refer to refined graphs with the respective clustering coefficient threshold.**



(a) Biological Process



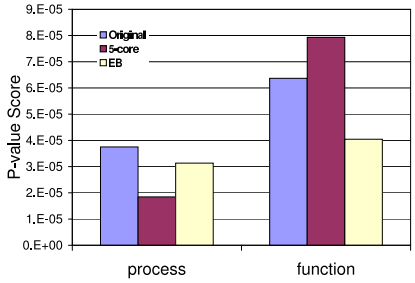
(b) Molecular Function

Figure 6: **Comparison of p-value scores before and after the Edge Betweenness refinement. P-value scores represent the average p-values of the partitions obtained using different partitioning algorithms.**

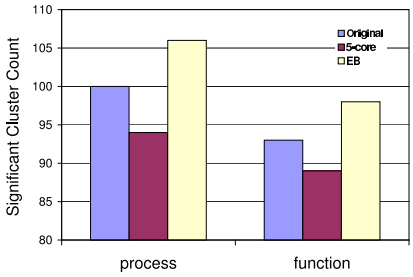
all three to test our refinement technique. Figure 6 shows the p-value scores for the two annotations before and after refinement using the different algorithms. As can be observed from the figure, the p-value scores are getting smaller (improving) in almost all the cases. The only exception to this is the molecular function p-value score of the spectral algorithm. However, multi-level algorithms can possibly produce better partitions than single-level versions. Multi-level form of the spectral partitioning algorithm improves scores for both ontologies.

4.2.3 Comparison of Edge Betweenness and Graph Cores Refinement Techniques: In this experiment, we compared the effectiveness of two refinement techniques. We ran the k-cores algorithm for $k = 5$ and the Edge Betweenness algorithm with the same size threshold ($T_{size} = 5$). We chose the size threshold to be five since dense components smaller than five may not correspond to a significant grouping. An optimal clustering coefficient threshold ($T_{cc} = 0.6$) is used for this experiment. Using the k-cores refinement, 23 of the 60 hubs are duplicated. The Edge Betweenness refinement allows 42 hubs to be duplicated with 34 of them duplicated only once and 8 of them duplicated twice.

The refined graphs obtained with k-core and Edge Betweenness refinement techniques as well as the original graph are partitioned using the kMETIS algorithm and the quality of the partitions are compared in terms of the p-value scores and the number of significant partitions. The results we obtained are depicted in Figure 8. As can be seen from Figure 8(a), 5-cores improves the p-value scores for biological process. On the other hand, the Edge Betweenness refinement algorithm is able to improve the p-value scores for both annotations. As depicted in Figure 8(b), 5-cores fail to increase the number of significant partitions whereas Edge Betweenness increases the number of significant partitions for both ontologies. These results suggest that Edge Betweenness refinement technique outperforms the k-cores although the quality of the partitions are improved by both methods. In addition, as we stated before, k-cores allows only a single duplicate for each hub, whereas Edge Between-

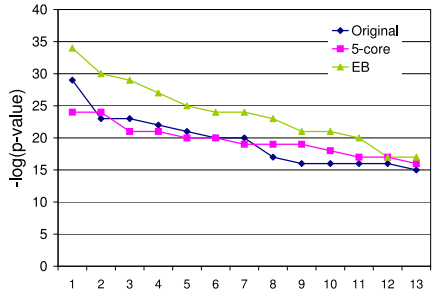


(a) P-value scores

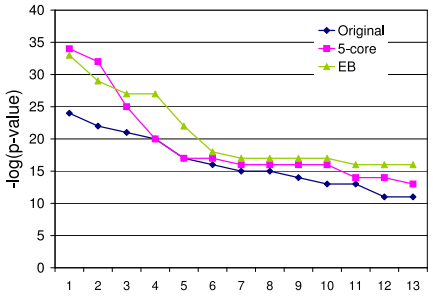


(b) Significant Cluster Count

Figure 7: **Comparison of k-cores and Edge Betweenness refinement techniques. 5-core represents k-cores and EB stands for Edge Betweenness.**



(a) Biological Process



(b) Molecular Function

Figure 8: **Best p-value scores obtained using kMETIS algorithm before and after the refinement. Top 10% of the partitions are compared. 5-core represents p-values obtained after k-core refinement, EB represents p-values obtained with Edge Betweenness refinement technique.**

ness allows as many duplications as there are dense subcomponents in the hub-induced subgraph.

Although the significance of improvement in p-value scores is considerable, the significance of this improvement is more obvious when we compare the best scoring partitions. In Figure 8 the top 10% of the p-value scores of the partitions is depicted. As seen from the figures, both ontologies are improving significantly (by as much as 10^9 in some cases). These results also indicate that the Edge Betweenness refinement technique performs better than the k-cores.

5 Discussion and Conclusion

In this paper, we have proposed two refinement techniques to improve modular decomposition of PPI graphs. We refined the PPI graph based on both graph cores and Edge Betweenness Partitioning algorithms. From our experimental results, we found that the Edge Betweenness algorithm is more suitable for this purpose. To partition the graph, we applied three partitioning algorithms and found the kMETIS algorithm to perform the best overall.

The experimental results that we obtained also indicate that duplicating the hubs of a Protein-protein interactions graph improves the modularity of the graph. Thus, we are able to obtain biologically significant partitions. A detailed examination of the partitions obtained revealed that the proposed method has three major benefits:

- Enhancement of available functional groupings: We obtain larger groups of proteins that are annotated with the same GO term from our refined graph than on the original graph.
- Isolation of additional functional groupings: We find groupings of proteins that could not be obtained from the original graph.
- Fuzzy-clustering: Our approach can find multi-functional proteins and group them into functional modules corresponding to each of their functions.

We now provide some illustrations from our results for each of these cases: KAP95 which is an essential protein is known to take part in 'nucleocytoplasmic transport'. Specifically, KAP95 (Karyopherin beta)

participates in a complex that mediates nuclear import of cargo proteins via a nuclear localization signal(NLS). It interacts with nucleoporins to guide transport across the nuclear pore complex [11]. When we partition the original DIP dataset, this protein is correctly grouped with proteins (NTF2, MLP2, SSA1, ASM4, YRB1, YNL253W, RNA1, NDC1) that are also annotated with 'nucleocytoplasmic transport' term with p-value 1.14e-08.

Using the Edge Betweenness refinement, KAP95 is duplicated once. The original hub and its duplicate appear in two separate partitions when we partition with the kMETIS algorithm. In one of these partitions, KAP95 is grouped with proteins (NTF2, SSA1, YRB1, RNA1, GSP1, SRM1, MTR10, KAP122, KAP142, KAP124, NUP157, NUP2, NUP1, NUP60, NUP82, NUP170, NUP145, NUP42) sharing the same biological process with p-value 1.07e-27. As can be seen from the protein names, the major difference between this group and the one from the original graph are the inclusion of NUPs(8 proteins) and KAPs(3 proteins) which are known as nucleoporins and karyopherins respectively. Karyopherins (including KAP95) are proteins that bind to their cargoes by recognition of specific nuclear localization signals or nuclear export signals. Transport through the nuclear pore complex is facilitated by transient interactions between the karyopherins and the nuclear pore complex proteins (referred to as nucleoporins) [22]. Thus, locating nucleoporins and karyopherins together is a noticeable benefit caused by the hub duplications. Clearly, our approach groups more proteins that belong to the same functional module together. This suggests that hub duplications make isolation of modules easier. These partitions are also valuable for predicting the functions of unknown proteins. In the group containing proteins that take part in 'nucleocytoplasmic transport', four proteins (YKL061W,YKR064W, YNL122C, YER004W) which do not have a known function also exist. Among these four, YKL061W is predicted to take part in 'nucleus-cytoplasm transport' process by Brun et al [6] in accordance with our findings. Since two different datasets and approaches are used to discover partitions, the overlap is noteworthy. This suggests that the other three proteins might have an unrevealed task in 'nucleocytoplasmic transport' process.

In addition to enhancing partitions, our method is also able to assign hub proteins which were originally in insignificant partitions into significant partitions. To illustrate this, we consider the hub protein LSM8. The LSM(Sm-like) proteins in Saccharomyces cerevisiae interact with each other and with U6 snRNA complex and they have an influence on pre-mRNA splicing [20]. In the original dataset, this protein is assigned to a partition which does not have any significant annotations either for biological process or molecular function. However, after the refinement, this protein is located into a partition which has a biological process annotation with p-value 1.2e-12. In addition to LSM8, ten other proteins (LSM2, LSM3, LSM5, PRP31, PRP21, PRP3, PRP4, PRP6, SMB1, SPP381) in this group are associated with biological process 'mRNA splicing'. LSM8 is located with the members of its complex (other SM-like proteins) as well as the components of U6 snRNP complex(PRP proteins). This example shows that our technique not only improves functional modules which can be identified from the original dataset, but also allows detection of functional modules which cannot be discovered from the original dataset.

Another advantage of our refinement technique is its ability to soft-cluster hub proteins into several partitions. This feature is important since some proteins are known to take part in multiple unrelated functional modules. CKA1 is one of these multi-faceted proteins and is involved in several cellular events. It is been reported to function in the maintenance of cell morphology and cell polarity, and in the regulation of the actin and tubulin cytoskeletons [8]. When the original DIP dataset is partitioned with kMETIS, CKA1 and seven other proteins (DEP1, RPD3, CDC73, RTF1, LEO1, CTR9, SPT16) that are annotated with 'transcription, DNA-dependent' term are located into the same partition. This combination is associated with p-value 3.47e-05. On the other hand, the Edge Betweenness technique duplicates CKA1 twice, so there exist three nodes corresponding to this protein. When we partition with kMETIS, all these nodes are assigned into different partitions resulting in three different groupings for protein CKA1. All three partitions, which we obtained by the use of hub-induced subgraphs, correspond to different functional modules of the CKA1 protein. One of these partitions is an enhancement of the 'transcription, DNA-dependent' functional module by associated p-value of 2.3e-19. In this partition, proteins (TAF3, TAF10, TAF11, TAF13, TAF14, CDC73, PAF1, SPT16, TFG1, CKA1, VPS36, SPT6, CTR9, FHL1, SPT5, ABF1, CHD1, IWS1, SAS3, VPS25, SNF8, LEO1, RTF1, CKB1, CKA2) are annotated with 'transcription, DNA-dependent' GO term in addition to CKA1 protein. The second partition in which CKA1 is located includes proteins (TPK1, TPK2, TPK3, CKA1, IRE1, PKH1). These proteins as well as the CKA1 protein have an annotated biological process term 'protein amino acid phosphorylation' with p-value 1.2e-05. The third partition contains proteins (CKA1, NOC3, NOP2, NOP4, NOP7, NOP15, NOP16, SPB1 TIF6,

SKI6, RPL5, YTM1, DBP10, RRP40, CSL4, DRS1, RRP6, DIS3, RPF2, RRP45, MAK11, RLP7) annotated for 'organelle organization and biogenesis' and associated with p-value 3.2e-12. Thus, we found that our technique, not only improved the obtainable partitions (by decreasing p-value from 3.47e-05 to 2.3e-19), but also grouped CKA1 with proteins that share its different functions.

Altogether these examples indicate the effectiveness of our approach on isolation of functional modules from the Protein-Protein interaction graphs.

In the future, we would like to extend this work by developing more effective techniques to improve modular decomposition of PPI graphs.

References

[1] *http://db.yeastgenome.org/cgi-bin/GO/goTermFinder*.

[2] A Abou-Rjeili and G Karypis. Multilevel algorithms for partitioning power-law graphs. *IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2006.

[3] R Albert, H Jeong, and A L Barabasi. Error and attack tolerance in complex networks. *Nature*, 406:378–382, 2000.

[4] M Ashburner and *et al.* Gene ontology: tool for the unification of biology. the gene ontology consortium. *Nat Genet.*, 25(1):25–29, May 2000.

[5] J Berg, M Lssig, and Andreas Wagner. Structure and evolution of protein interaction networks: a statistical model for link dynamics and gene duplications. *BMC Evolutionary Biology*, 4:51, 2004.

[6] C Brun, F Chevenet, D Martin, J Wojcik, A Gunoche, and B Jacq. Functional classification of proteins for the prediction of cellular function from a protein-protein interaction network. *Genome Biology*, 5, 2003.

[7] C Brun, C Herrmann, and A Guenoche. Clustering proteins from interaction networks for the prediction of cellular functions. *BMC Bioinformatics*, 5(95), July 2004.

[8] D A Canton and D W Litchfield. The shape of things to come: An emerging role for protein kinase ck2 in the regulation of cell morphology and the cytoskeleton. *Cell Signalling*, 18:267–275, 2006.

[9] F Chung, L Lu, T G Dewey, and D J Galas. Duplication models for biological networks, 2002.

[10] P Crucitti, V Latora, M Marchiori, and A Rapisarda. Error and attack tolerance of complex networks. *Physica A*, 340:388–394, 2004.

[11] D Gilchrist and M Rexach. Molecular basis for the rapid dissociation of nuclear localization signals from karyopherin alpha in the nucleoplasm. *J. Biol. Chem*, 278:51:51937–51949, 2003.

[12] J D Han and *et al.* Evidence for dynamically organized modularity in the yeast protein-protein interaction network. *Nature*, 430(6995):88–93, 2004.

[13] S Hauck and G Borriello. An evaluation of bipartitioning technique. *In Proc. Chapel Hill Conference on Advanced Research in VLSI*, 1995.

[14] B Hendrickson. Graph partitioning and parallel solvers: Has the emperor no clothes? *In Proc. Irregular98*, pages 218–225, 1998.

[15] B Hendrickson and R Leland. The chaco users guide, version 2.0. technical report sand94-2692, sandia national laboratories. 1994.

[16] J Hua, D Koes, and Z Kou. Finding motifs in protein-protein interaction networks. *Project Final Report, CMU*, 2003.

[17] H Jeong, S P Mason, A L Barabasi, and Z N Oltvai. Lethality and centrality in protein networks. *Nature. 411:44.*, 411:41–42, 2001.

[18] G Karypis and V Kumar. Unstructured graph partitioning and sparse matrix ordering system. technical report. *http://www-users.cs.umn.edu/ karypis/metis/metis/files/manual.pdf*.

[19] J Lin, J Qian, D Greenbaum, P Bertone, R Das, N Echols, A Senes, B Stenger, and M Gerstein. Genecensus: genome comparisons in terms of metabolic pathway activity and protein family sharing. *Nucleic Acids Res*, 30:4574–82, 2002.

[20] A E Mayes, L Verdone, P Legrain, and J D Beggs. Characterization of sm-like proteins in yeast and their association with u6 snrna. *EMBO J.*, 18(15):4321–4331, 1999.

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

[22] L F Pemberton and B M Paschal. Mechanisms of receptor-mediated nuclear import and nuclear export. *Traffic*, 6:187, 2005.

[23] A L Barabsi S Yook, Z N Oltvai. Functional and topological characterization of protein interaction networks. *Proteomics*, 4:928–942, 2004.

[24] S B Seidman. Network structure and minimum degree. *Social Networks*, 5:269–287, 1983.

[25] H D Simon and S Teng. How good is recursive bisection? *Technical Report RNR-93-012, NAS Systems Division*, 1993.

[26] A Thomas, R Cannings, N A M Monk, and C Cannings. On the structure of protein-protein interaction networks. *Biochemical Society Transactions*, 31:1491–1496, 2003.

[27] A Vazquez, A Flammini, A Maritan, and A Vespignani. Modeling of protein interaction networks. *Complexus*, 1:38, 2003.

[28] D Watts and S Strogatz. Collective dynamics of small world networks. *Nature*, 393(6684):440–442, June 1998.

[29] L F Wu, T R Hughes, A P Davierwala, M D Robinson, R Stoughton, and S J Altschuler. Large-scale prediction of saccharomyces cerevisiae gene function using overlapping transcriptional clusters. *Nature Genetics*, 31:255–265, June 2002.