A Matrix-based Multilevel Approach to Identify Functional Protein Modules

Suely Oliveira[†] and Sang-Cheol Seok[†]

[†]Department of Computer Science, The University of Iowa, Iowa City, IA 52242, U.S.A. E-mail:{oliveira,sseok}@cs.uiowa.edu

Abstract: Identifying functional modules is believed to reveal most cellular processes. There have been many computational approaches to investigate the underlying biological structures [2, 6, 15, 19, 22, 23]. A spectral clustering method plays a critical role identifying functional modules in a yeast protein-protein network [19]. We present an unweighted-graph version of a multilevel spectral algorithm which more accurately identifies protein complexes with less computational time.

Keywords: Functional modules; protein-protein interaction network; spectral algorithms; multilevel methods.

Reference to this paper should be made as follows: Suely Oliveira and Sang-Cheol Seok. (xxxx) 'A Matrix-based Multilevel Approach to Identify Functional Protein Modules', *Int. J. Bioinformatics and Research applications*, Vol. x, No. x, pp.xxx–xxx.

Biographical Notes: Suely Oliveira is an Associate Professor in the Computer Science Department at the University of Iowa. At the University of Iowa she has also holds a courtesy appointment in Mathematics and is a member of the Program in Applied Mathematics and Computational Sciences. She received her PhD from the University of Colorado in 1993. She was at the Australian National University from 1993 to 1994, and at Texas A&M University from August 1994 to May 1998 as an Assistant Professor.

Dr. Oliveira has organized many workshops and minisimposia, and is one of the editors of Electronic Transactions of Numerical Analysis special issue in Combinatorial Scientific Computing. She has published over 50 articles in the areas of numerical analysis, parallel algorithms, scientific computing, combinatorial scientific computing, and is co-author of the book "Writing Scientific Software" to be published by Cambridge Press University, Fall 2006.

Sang-Cheol Seok received his B.S. and M.S. degrees in mathematics at the Pusan National University, Pusan, Korea. He is currently working toward Ph.D. degree in computer science at the University of Iowa.

His research interests include multilevel methods in document clustering and identifying functional modules in protein-protein interaction networks.

1 Introduction

Most cellular processes are carried out by groups of proteins. Identifying functional modules in protein-protein networks is considered as one of the most important and challenging research topics in computational systems biology. There have been many recent computational approaches to disclose the underlying biological structures [2, 6, 15, 19, 22, 23]. These approaches are divided into two groups. One group uses machine learning approaches to construct weighted graphs by integrating existing data sets to predict protein complexes [15, 23]. Another group tries to extract highly connected subgraphs or dividing a whole network into groups of clusters on a protein-protein interaction (PPI) network [2, 6, 19, 22].

There are a number of challenges in treating protein-protein interaction data. One is that many high-throughput experiments have high error rates, which results in a great many false positives for interactions between proteins. Another challenge is that some proteins are "utility" proteins that interact with very large numbers of other proteins; these might, for example, provide common services to many different parts of the cell. The former challenge can make accurate identification of functional modules difficult, while the latter challenge tends to make the entire proteome appear to be a single indivisible functional module.

[19] propose a two-level architecture for a yeast proteomic network. They construct a smaller network from a PPI network by removing proteins which interact with too many or too few proteins. Removing proteins with too few interactions can remove many of the effects of false positives. On the other hand, removing proteins with the largest numbers of interactions can make the finer structure of the interactions more evident. A clustering algorithm is applied to this residual network. Validation of clusters is performed by comparing the clustering result with a protein complex database, the Munich Information Center for Protein Sequences (MIPS). A spectral clustering method plays a critical role for identifying functional modules in the PPI network in their research.

Recently, we successfully applied a multilevel spectral algorithm to cluster a group of documents using similarity matrices which are mostly dense with entries between 0 and 1 [18]. Like large-scale networks, the vertex connectivities of proteomic networks follow a scale-free power-law distribution. That means that, the proteomic network consists of a small number of high degree nodes and a majority of low degree nodes. However, the proteomic network has no edge weights. In this paper, we present an unweighted-graph version of a multilevel spectral algorithm which identifies more protein complexes with less computation time than the basic spectral approach [18].

Multilevel algorithms have a long history, mostly for partial differential equations in numerical analysis but also for graph partitioning, such as in METIS [12]. Recently, multilevel schemes have been applied to graph clustering [6, 18]. MultiLevel (ML) algorithms conventionally consist of three main steps: coarsening, partitioning, and decoarsening. Multilevel clustering algorithms, like multilevel partitioning algorithms, mostly try to improve existing clustering algorithms using good coarsening or matching algorithms. ML algorithms not only improve the quality of the clustering, but also significantly reduce computational time.

The best well-known ML algorithms include *random-edge matching* (REM) and *heavy-edge* matching (HEM) for coarsening [13]. There are more recent matching

algorithms like the Linear-time Approximation to Maximal matching (LAM) algorithm of [16]. These algorithms are mainly designed for weighted graphs. In [18] we compared two different coarsening algorithms for weighted graphs. We showed that when nodes are merged in order of decreasing edge weight we can expect better results. We called this Sorted Matching (SM), because each edge weight represents how close two nodes are. In the PPI networks the unweighted graph cases can not use any of these edge-oriented methods directly because all edge weights are initially one. So in this paper we introduce the Heavy-Edge-Small-Node (HESN) heuristic. We will show that the HESN algorithm outperforms a random matching algorithm and a matching algorithm which focuses on maximizing the number of nodes collapsed. Our algorithm also has better computational performance than an ML algorithm shown to work very well on power-law networks.

We also introduce *diagonal-edge weighting* for unweighted graphs. The spectral clustering algorithm in this research uses similarities between vertices. Unweighted graphs have uniform weights for edges, that is, the same similarities. In a weighted graph each node has the highest similarity with itself. To expect the same effect we add self edges to each node with the same weight as the degree of the node. This weighting process contributes to our clustering algorithm, especially for the refining step.

The remainder of the paper is organized as follows. In Section 2, we present an overview of ML spectral algorithms and talk about important features of PPI networks in Pothen et al.'s two-level approach. In Section 3, we present theoretical insights of ML spectral clustering. Two model networks used in this paper are described in Section 4. One of them is used to apply ML algorithms and the clustering results are compared with the other model network. Various ML matching algorithms are described and discussed in Section 5. Computational experiments are presented in Section 6 showing that our ML algorithm outperforms current greedy-based approaches.

2 Background on Multilevel Approaches and Clustering Algorithms

Let $G = (V, E)$ be a graph with vertex set V and set of undirected edges E. One of the most commonly used data structures for graphs are matrices. Matrix representations are very useful to store weights for edges and vertices. We can also use many well-known computational techniques from Linear Algebra. In our matrix representations $S = (s_{ij})$, diagonal entries s_{ii} store the weights of vertices and off-diagonal entries s_{ij} represent edge weights. Given interaction data, we typically set the vertex weights to one and $s_{ij} = 1$ if there is an interaction between proteins i and j, with $s_{ij} = 0$ otherwise. Our ML algorithms use this matrix representation.

2.1 Multilevel Clustering

The basic concept of "Multilevel clustering" algorithms is that when we have a large graph $G = (V, E)$ to partition, we construct a smaller graph whose vertices are groups of vertices from G. We can apply a clustering method to this smaller graph, and transfer the partition to the original graph. This idea is very useful because smaller matrices or graphs require much less time to cluster. The process of

constructing the smaller matrix is called coarsening, and the process of transferring the partition is called decoarsening.

Coarsening and decoarsening steps are implemented by multiplying a graph matrix S by a special coarsening matrix C. Each entry of C is either 0 or 1. We set $c_{ij} = 1$ if node i of the fine graph belongs to node j of the coarsened graph. How to construct C is decided by matching algorithms explained in section 5. A series of matrices $S_0, S_1, \cdots, S_l$ are recursively constructed using $C_1, \cdots, C_l$ in the form of $S_i = C_i' \cdot S_{i-1} \cdot C_i$ with $i = 1, \cdots, l$. Note that C' is the transpose of C (i.e. $c_{ij} = c_{ji}$) . A partitioning algorithm is applied to matrix S_l and we will have an initial partition Cut in the coarsest level.

Partitioning algorithms fall into two categories: direct partitioning and recursive bipartitioning (divisive partitioning). A recent clustering algorithm is spectral partitioning. Direct spectral partitioning algorithms compute $k-1$ eigenvectors of the graph Laplacian and use them to distinguish k clusters directly (see section 2.2). Recursive bipartitioning algorithms repeatedly performs two main steps. One is selecting a cluster to split, and the other is applying a two-way clustering algorithm. Two-way spectral algorithms try to find a pair of disjoint subsets (A, B) of V by computing the eigenvector q_2 associated with the second smallest eigenvalue of the graph Laplacian. Divisive algorithms recursively choose a cluster which satisfies a selection criterion and divide it, until we have the predefined number of clusters or until all current clusters satisfy a certain condition. On level i we have a partition (A_j) of the vertices of G_i. To represent the partition, we use a vector Cut_i on level i where $(Cut_i)_k = j$ if $k \in A_j$.

Decoarsening is how we get back to the original graph. The partition from the coarsest level is mapped into finer levels by using a proper coarsening matrix $Cut_i = C_i' \cdot Cut_{i-1}$ where i is the level number of the coarser level. Then a Kernighan-Lin (KL) type refinement algorithm is applied to improve the quality at each level ([14]). KL starts with an initial partition; it iteratively searches for nodes from each cluster of the graph where moving a node to one of the other clusters leads to a better partition. For each node, there may be more than one cluster to give smaller objective function values than the current cut. So the node moves to the cluster that gives the biggest improvement. The iteration terminates when it does not find any node to improve the partition.

2.2 Spectral Algorithms

The best known spectral partitioning algorithms are RatioCut, NormalizedCut and MinMaxCut algorithms [11, 20, 8]. Two-way spectral clustering algorithms start by constructing optimization problems. These three problems are described as follows.

- MinMaxCut: Minimize

$$J_{MMC}(A, B) = \frac{s(A, B)}{s(A, A)} + \frac{s(A, B)}{s(B, B)} = \frac{s(A, \bar{A})}{s(A, A)} + \frac{s(B, \bar{B})}{s(B, B)}, \tag{1}$$

- RatioCut: Minimize

$$J_R(A,B) = \frac{s(A,B)}{|A|} + \frac{s(A,B)}{|B|} = \frac{s(A,\bar{A})}{|A|} + \frac{s(B,\bar{B})}{|B|}, \tag{2}$$

- NormalizedCut: Minimize

$$J_N(A,B) = \frac{s(A,B)}{s(A,A)+s(A,B)} + \frac{s(A,B)}{s(B,B)+s(A,B)} \tag{3}$$

$$= \frac{s(A,\bar{A})}{s(A,A)+s(A,\bar{A})} + \frac{s(B,\bar{B})}{s(B,B)+s(B,\bar{B})}, \tag{4}$$

where $s(A,B) = \sum_{i\in A, j\in B} s_{ij}$ and $\overline{A} = \{\, i = 1, 2, \ldots, n \mid i \notin A \,\}$.

In [8], a continuous approximation to this problem has the solution which is the eigenvector q_2 associated with the second smallest eigenvalue of the system $(D-S)q = Dq$, where $D = diag(d_1, d_2, \cdots, d_n)$ and $d_i = \sum_j s_{ij}$. The partition (A,B) is calculated by finding index *opt* such that the corresponding objective function gets optimum value with the partition, $A = \{q_2(i) \mid q_2(i) < q_2(opt)\}$ and $B = \{q_2(i) \mid q_2(i) \geq q_2(opt)\}$.

Objective functions for two-way partitioning optimization formulations can be easily generalized for K-way partitioning such as

$$J_{MMC}(A_1, \ldots, A_k) = \sum_{i=1}^{k} \frac{s(A_i, \bar{A}_i)}{s(A_i, A_i)}. \tag{5}$$

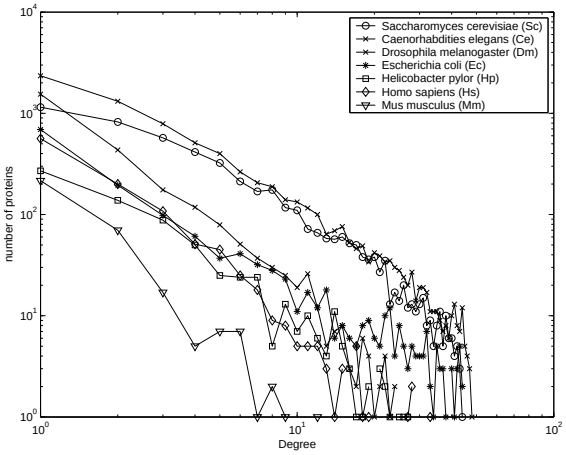
Direct partitioning algorithms compute $k-1$ eigenvectors, $q_2, \cdots, q_k$, of the same generalized eigenvalue system for two-way and then use (5) to create k clusters instead of (1) to create two clusters. This objective function (5) is also used for recursive bipartitioning during the refinement phase.

The optimum value of two-way MinMaxCut is called the cohesion of the cluster and can be an indicator to show how closely vertices of the current cluster are related [8]. This value can be used for the cluster selection algorithm. The cluster which has the least cohesion is chosen for partitioning. Another selection criterion considered in this research is the average similarity, $\bar{s}_i := s(A_i, A_i)/|A_i|^2$. Both algorithms were reported earlier to work very well [8].

2.3 Features of Interaction networks and Two-level Approach

Features of Proteomic networks: Graph theory is commonly used as a method for analyzing PPIs in Computational Biology. Each vertex represents a protein, and edges correspond to experimentally identified PPIs. Proteomic networks have two important features [4]. One is that the degree distribution function $P(k)$ (the number of nodes with degree k) follows a power law $P(k) \approx \text{constant}\, k^{-\alpha}$ (and so is considered a scale-free network). This means that, most vertices have low degrees, called low-shell proteins, and a few are highly connected, called hub proteins. Figure 1 shows the degree distributions of 7 organisms in DIP database (See section 4.1). The other feature is the *small world* property which is also known as *six degrees of separation*. This means the diameter of the graph is small compared with the number of nodes.

Figure 1 The degree distribution of 7 organisms in DIP database.



The standard tools to understand these networks are the clustering coefficient (Cc), the average path length, and the diameter of the network. The clustering coefficient (Cc_i) is defined in terms of $E(G(v_i))$, the set of edges in the neighborhood of v_i. The clustering coefficient is the probability that a pair of randomly chosen neighbors of v_i are connected. That is,

$$Cc_i = \frac{2}{deg(v_i)[deg(v_i) - 1]} \cdot |E(G(v_i))| \tag{6}$$

where $G(v_i)$ is the neighborhood of v_i, and $E(G(v_i))$ the edges in $G(v_i)$.

Since the denominator is the maximum possible number of edges between vertices connected to v_i, $0 \leq Cc_i \leq 1$. The global clustering coefficient can be simply the average of all individual clustering coefficients (Cc_i) like

$$\overline{Cc} = \sum_{i=1}^{n} Cc_i/n. \tag{7}$$

But this "average of an average" is not very informative [4]; one alternative is to weight each local clustering coefficients

$$Cc = \sum_{i=1}^{n} \frac{deg(v_i)}{MaxDeg} Cc_i/n, \tag{8}$$

where $MaxDeg$ is the maximum degree in the network. We use the latter Cc as our clustering coefficient for the rest of the paper.

The path length of two nodes v_i and v_j is the smallest sum of edge weights of paths connecting v_i and v_j. For an unweighted network, it is the smallest number of edges to go through. The average path length is the average of path lengths of all pairs (v_i, v_j). The diameter of the network is the maximum path length. In Section 4, we talk about these features with our model networks in more details.

The advantage of the two-level approach: The two level approach was proposed by [19] The main idea is derived from the k-cores concept which was introduced by V. Batagelj and M. Zaversnik in graph theory [3]. If we repeatedly

remove vertices of degree less than k from a graph until there are no longer any such vertices, the result is the k-core of the original graph. The vertices removed are called the low-shell vertices. The two-level approach pays attention to three facts in protein-protein interaction networks:

- The hub proteins have interactions with many other proteins, so it is hard to limit them to only one cluster and the computational complexity increases when they are included.
- There are many low-shell proteins, which increases the size of network. These nodes are easy to cluster when the nodes they are connected to are clustered first.
- Proteomic networks are mostly comprised of one big component and several small components.

So, disregarding hub proteins and low-shell proteins, and confining attention to the biggest component of proteomic networks leaves us to focus on the nodes which are most meaningful to cluster. Another advantage of this approach is that this architecture favors the size of the residual network which requires less computational time.

3 New Theoretical Insights about Multilevel Spectral Clustering

All spectral algorithms have objective functions (also called criterion functions) in certain optimization formulations. We saw three well-known spectral clustering approaches in the previous section. The theoretical aspect of our ML algorithms is discussed in this section. The original graph (the finest graph) and the coarsest graph are connected by a series of coarsening matrices as we saw in Section 2.1. The question which must be addressed is if the ML strategy changes the optimum structure of clusters; that is, whether or not the optimum values are changed during the coarsening and decoarsening processes.

Assume S' is the matrix coarsened from S with a coarsening matrix C, $S' = C^T \cdot S \cdot C$. The similarity within clusters $s(A, A)$ of section 2.2 can be represented by

$$s(A, A) = \sum_{p,q \in A} s_{pq} = e_A^T S e_A, \tag{9}$$

where e_A is a vector which has 1's for non-zero elements of A and 0's for the rest. We first show that $Ce_{A'} = e_A$, where A' is the coarse graph set which generates A. For any i,

$$(Ce_{A'})_i = \sum_p C_{ip}(e_{A'})_p \tag{10}$$

$$= \sum_{p \in A'} C_{ip}. \tag{11}$$

Since $\sum_{p \in A'} C_{ip}$ is 1 if i is in some $p \in A'$ and 0 otherwise, $Ce_{A'} = e_A$. Note that for each i there is only one p where $E_{ip} = 1$. i.e. each fine grid node belongs to only one coarse grid node. Thus $(Ce_{A'})_i = (e_A)_i$

Lemma 3.1. *If A' and B' are sets of coarse graph nodes, and A,B the sets of fine grid nodes they generate, then $s(A,B) = s(A',B')$.*

Proof

$$s(A',B') = e_{A'}^T S' e_{B'} = e_{A'}^T C^T S C e_{B'} \tag{12}$$

$$= (Ce_{A'})^T S (Ce_{B'}) = e_A^T S' e_B = s(A,B) \tag{13}$$

Theorem 3.2. $J_{MMC} = s(A,B)/s(A,A)+s(A,B)/s(B,B) = s(A',B')/s(A',A')+ s(A',B')/s(B',B')$.

Proof Applying lemma 3.1 to clusters A and B gives us that $s(B',B') = s(B,B)$ and $s(A',B') = s(A,B)$.

The same result holds for NormalizedCut J_N of (4). However, RatioCut J_R of (2) does not preserve function values because the number of nodes is different in a cluster after coarsening or decoarsening. The corresponding results hold for K-way objective functions. This theorem shows that objective function values of J_{MMC} and J_N do not change as levels go up and down in the all ML approaches.

4 Model Networks

There are databases which contain protein-protein interactions as well as cellular localization, gene regulation and the context of these interactions. The best known are KEGG (www.genome.ad.jp/kegg), BIND (www.bind.ca), MIPS (mips.gsf.de), PROnet (www.pronet.doubletwist.com) and DIP (dip.doe-mbi.ucla.edu). We consider two protein-protein interaction databases. One, the Database of Interacting Proteins (DIP), provides protein-protein interactions in over 100 organisms including yeast, and the other, the Munich Information center for Protein Sequence (MIPS), has a list of experimentally trusted functional modules in yeast. We mostly use the proteomic network for yeast in the first database and compare with the functional modules of the second (MIPS) to evaluate the clustering results.

4.1 Database of Interacting Proteins

DIP is a protein interaction database. The protein-protein interactions can be detected by high-throughput experiments such as yeast two-hybrid assays (Y2H) and Mass Spectrometry Tandem Affinity Purification (MS-TAP). As of December 2005, the database catalogs 55,000 unique interactions among more than 19,000 proteins from over 62,000 distinct experiments and over 3,000 data sources. Each protein included in DIP dataset is experimentally determined and cross-referenced to at least one of the major sequence databases (SWISS-PROT, GenBank, PIR).

One interaction network is for the budding yeast, *Saccharomyces cerevisiae*, that is considered the simplest and so the most investigated organism. They also provide a smaller dataset called CORE which contains the pairs of interacting proteins identified that were validated according to the criteria described in [5].

Pothen et al. presented a two-level architecture on this CORE dataset [19]. The network has 2610 proteins and 6236 interactions. Their idea is that removing high degree proteins (called hub proteins) and low degree proteins (low-shell proteins) from the network before clustering leads to a better partitioning and then the removed nodes can be added to the partitioning. The residual network after removing hub proteins and low shell proteins has 499 proteins and 1229 interactions.

Instead of using the small network (CORE dataset), we use the DIP network which has 4931 nodes and 17471 edges to validate our ML algorithms. Constructing a residual network starts by removing nodes that have degree 20 or more from the original network. Then low-shell proteins whose degree is 3 or less are pruned from the biggest component. The residual network has 1078 nodes and 2778 edges.

All 7 protein-protein interaction networks we use in this research are presented in *Table 1*. The first rows for each organism are about the original networks and the second rows have the information for their residual networks. They all have different numbers of proteins and interactions. More importantly they have different degree distributions. So, we use different thresholds for hub proteins and low-shell proteins. Hub proteins and low-shell proteins are defined as all proteins with degrees of 20 or greater and less than 3 for *Drosophila melanogaster* and yeast. For the other organisms, hub proteins and low-shell proteins have degree of 30 or greater and less than 2. The size of the residual network is closely related to the clustering coefficient, average path length and diameter of the original network. For example, the residual network of *Caenorhabditis elegans* is much smaller than that of *Escherichia coli* even though the original network of *Caenorhabditis elegans* has more proteins because the clustering coefficient of *Caenorhabditis elegans* is much smaller and average path length and diameter are longer and bigger. That is, *Caenorhabditis elegans* has less cohesive network.

Note that original networks have edges connecting the same vertices and all these are disregarded in this research because these edges have nothing to do with clustering. This self edge is different from the diagonal-edge weighting introduced in Section 6.3. *Drosophila melanogaster* is a fruit fly and *Saccharomyces cerevisiae* is baker's yeast.

Organism	Proteins	Interactions	Cc	Avg. path	Diameter
Drosophila melanogaster	7451	22636	0.00044	4.39	11
(fruit fly)	1050	2298	0.00557	5.16	9
Saccharomyces cerevisiae	4919	18224	0.0027	4.14	11
(yeast)	1078	2778	0.0735	5.15	10
Escherichia coli	1640	5821	0.0075	3.72	11
(bacterium)	377	735	0.0321	4.89	11
Caenorhabditis elegans	2638	3970	0.00044	4.80	14
(nematode/worm)	15	26	0.333	1.65	3
Helicobacter pylori	710	1359	0.0031	4.13	9
(bacterium)	356	816	0.0045	3.84	8
Homo sapiens	1065	1318	0.0092	6.80	21
(human)	257	546	0.0768	6.28	15
Mus musculus	329	274	0.00044	3.56	9
(mouse)	12	17	0.264	2.83	6

Table 1 The numbers of proteins and interactions to obtain protein-protein interaction networks and residual networks with network analyzing features in 7 organisms in DIP.

4.2 Database of Functional Modules

MIPS contains the molecular structure and functional networks and data of yeast for comparative analysis. The currently annotated functional modules for yeast have 6451 proteins in all with 4022 proteins with known gene-id and 2429 proteins with unknown gene-id. 800 proteins of them are included in the residual network of the DIP protein-protein network as well.

Many functional modules in the yeast are found in a hierarchy of as many as four levels. That is, some proteins are related to only one cellular process and some proteins work in a group and groups of them are involved in another cellular process. For example, proteins in the Anaphase promoting complex (id:60) are not involved in any other cellular process. However, proteins in the Dynactin complex (id:140.30.30.30) cooperate with proteins in Kinesin-related motorproteins (id:140.30.30.10) and Dynein-complex motorproteins (id:140.30.30.20) to make a bigger complex: Tubulin-associated motorproteins. Similarly, all proteins in complexes whose id start with 140 are included in the highest complex Cytoskeleton (id:140). Let us call these smallest functional units leaf modules.

5 Coarsening Algorithms

A matching in a graph is a set of edges in which no two of them are incident on the same node. A matching is maximal when any edge in the graph that is not in the matching has at least one of its endpoints matched. Some algorithms aim to match as many nodes as possible and some aim to maximize the sum of all edge weights [21, 10]; this is often referred to as maxi*mum* matching. These algorithms are too time intensive and designed for weighted graphs [16]. Our new matching algorithm tries to find a compromise.

The simplest matching for unweighted graphs is random matching. One node is randomly visited and one of unmatched node is randomly chosen to be merged with the node (Random Vertex Random Merge, RVRM). A drawback is that the nodes with low degrees have higher chance to be left unmerged than high degree nodes. In order to avoid this problem we can pick the lowest degree node among unmerged nodes and choose one of the unmerged nodes randomly to merge (Lowest degree Vertex Random Merge, LVRM). Thus this algorithm tends to merge more nodes than RVRM.

We define the weights of edges as follows. The edge weights are all 1's to start with, but become the sum of the number of edges combined after one matching step. A node weight is defined as the total number of nodes merged into it.

When we have different edge weights we can apply Sorted Matching (SM) idea for coarsening in subsequent levels. SM was used earlier by us for weighted graphs and merged nodes in order of decreasing edge weight. In the PPI network, at first we have equal edge weights. We perform the first level of coarsening by combining nodes with each other, as long as they are not matched. The results are similar for any order we pick up for this step. After this matching we will have groups of edges which share the same weight (the maximum resulting edge weight will be 4 for a clique with 4 nodes/vertices). We consider these groups in order of decreasing edge weight. Within each group we give higher priority to the edge with lower combined

node weights, i.e. we take the edge with maximum $1/w(n_i)+1/w(n_j)$ as a tie-break rule, where $w(n_i)$ and $w(n_j)$ are the node weights, that is, the number of nodes, of supernodes n_i and n_j. We call this matching scheme Heavy-Edge-Small-Node (HESN).

Karypis et al. also present heuristic matching-based algorithms for unweighted networks [1]. Their heuristic algorithms were tested for power-law networks. Various algorithms are presented and experimentally compared. They reported that fewer within cluster edges are expected to be deleted when vertices are visited randomly and one of the unmerged nodes is chosen in a greedy strategy. We include one of their coarsening algorithms in this research and compare it with ours. This Random Visit-and-Merge Greedy (RVMG) algorithm visits nodes randomly and a local greedy strategy is applied. In their greedy strategy, vertices that have connected edges with bigger edge weights are considered first. Smaller node weights are used to break ties for the same edge weights.

6 Computational Experiments

Computational results comparing the four different matching algorithms for unweighted graphs described in the previous section are presented first. Then we show how three spectral algorithms work on proteomic networks. The impact of the ML approach is discussed for different levels. We also show that diagonal-edge weighting improves the current ML spectral algorithms. Two measures are used to validate our computational results. One is the triplet, κ. The first item of κ is the total number of edges within clusters, the second is number of edges between clusters. The last item is the maximum number of edges between any two clusters. While this measure does not in itself indicate the quality of the clustering, it does provide some insight: The larger the first number and the smaller the second and third, the more cohesive clusters a network has. The other measure we use is the number of nodes correctly clustered according to the MIPS database. Each cluster formed from the DIP residual network is compared with the leaf functional modules listed in MIPS. We define τ as the sum of the maximum number of correctly matched proteins of each cluster.

One property of τ is that it increases as the number of clusters increases because the sizes of supernodes decrease.

6.1 Computational Comparison of Greedy-based Algorithms

Table 2 shows the comparison of four coarsening algorithms without partitioning and refinement. This table reveals the quality of supernodes at the coarsest level. The total numbers of correctly matched proteins, that is, how many proteins are correctly grouped in the same supernodes, are listed after various levels of coarsening. The first row for each algorithm has the sizes of the graphs (number of nodes) at the various levels of our ML scheme. LVRM results in the smallest graph in coarsest level than other algorithms. But HESN and RVMG generate more cohesive clusters as the second row of each coarsening on this table shows. That is, there are more edges inside clusters and less edges crossing them. Moreover, more proteins match MIPS dataset in the third rows by HESN.

# of levels	3	4	5	
RVRM	607-340-185 (1036,1742,17) 459	612-346-192-105 (1216,1562,22) 346	616-339-184-100-53 (1283,1495,41) 218	size of graph κ τ
LVRM	573-296-151 (1062,1716,18) 419	573-302-157-79 (1226,1552,17) 282	571-302-156-79-40 (1254,1524,23) 182	size of graph κ τ
RVMG	607-332-177 (1284,1494,30) 489	602-331-178-93 (1527,1251,21) 391	603-334-182-96-50 (1673,1105,11) 288	size of graph κ τ
HESN	601-333-182 (1351,1427,37) 514	601-333-182-102 (1602,1176,11) 397	601-333-182-102-56 (1749,1029,8) 295	size of graph κ τ

Table 2 Comparison of different coarsening algorithms with different levels. First row of each algorithm has the numbers of nodes of graphs, starting with the original and followed by numbers of nodes for the coarsened graphs.. Second rows have the number edges within and between clusters, and maximum number of edges between clusters (κ). Third rows have the number of correctly grouped nodes (τ).

Figure 2 The number of correctly clustered proteins to form 40, 50, and 60 clusters when four different coarsening algorithms are applied with various levels with MinMaxCut partitioning.

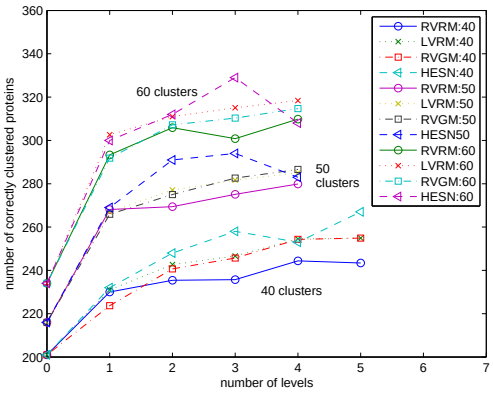


Figure 2 shows the performance of four coarsening algorithms when MinMaxCut partitioning and KL refinement algorithms are actually applied. Since RVRM, LVRM, and RVMG use random algorithms, we averaged 10 experiments for each method. HESN mostly clusters more proteins correctly than other three coarsening algorithms. Furthermore, we conclude that using ML clustering algorithms results in better clustering results with any of the coarsening algorithms as Figure 2 shows. For the computational experiments in the remaining of this paper we will use HESN as our coarsening algorithm, because of its better performance.

# of levels	0	1	2	3
RatioCut	(1768,1010,123) 181 366.4	(1721,1057,24) 216 68.7	(1665,1113,16) 254 50.2	(1712,1066,19) 243 42.5
NormalizedCut	(1544,1234,29) 254 298.1	(1452,1326,36) 225 60.1	(1712,1066,11) 283 21.4	(1630,1148,25) 254 16.9
MinMaxCut	(1624,1154,18) 271 333.4	(1772,1006,20) 310 158.4	(1823,955,26) 310 94.5	(1883,895,40) 319 104.1

Table 3 Comparison of three spectral algorithms with different number of levels to form 60 clusters. For each spectral method the first line shows κ, the second line shows the number of correctly grouped nodes, and the third line shows the total time consumed (in seconds).

no. of clusters	40	60
no. weighting	151 (1296,1482,209)	224 (1198,1580,86)
Adding 1	192 (1596,1182,50)	226 (1477,1301,37)
Adding degree	201 (1652,1126,50)	234 (1490,1288,46)

Table 4 Comparison of three different weighting methods on diagonal entries of S with 40 and 60 clusters. The entries show the number of correctly clustered proteins (τ) and the edge information (triplet κ) in parenthesis.

6.2 Comparison of Spectral Algorithms

The comparison of three spectral algorithms for forming 60 clusters is shown in *Table 3*. HESN is used as the coarsening algorithm. All nodes are weighted with degrees (see Section 6.3). Average similarity cluster selection algorithm is used for this comparison. NormalizedCut takes the least time with any levels. However, MinMaxCut mostly gives better clustering results. One thing we notice from this table is that MinMaxCut gives not only the minimum cuts but also the maximum edges within clusters.

6.3 Various Levels and Identifying Functional Modules

One issue when we use MinMaxCut spectral clustering algorithm, (see (1)), is that it may have zero for denominators because all diagonal entries of S are zero. One way to avoid this case is to compute the objective function values without including the clusters whose within similarity $s(A, A)$ is zero. Instead, we tried setting $s_{ii} = 1$ and tried setting $s_{ii} = \deg(v_i)$. To compare these three algorithms, we generated two groups of clusters without ML algorithms. *Table 4* shows that we expect better results when we set the diagonal entries of S to be the degrees of the nodes.

Table 5 shows the effect of our ML algorithm on finding functional modules. We considered the ML algorithms with 0 through 3 levels and use HESN and 60 clusters in all experiments for this table. As shown in this table, the number of correctly clustered proteins (τ) increases and the edge connectivities (κ) become better as the number of levels increases. The last row of the table shows the total timing and the fourth row shows the timings for the 3 parts of ML (coarsening, partitioning and decoarsening). The total time consumed decreases up to 3 levels and then

# of levels	0	1	2	3
no. of nodes	1078	601	333	182
edge info (κ)	(1490,1288,46)	(1769,1009,31)	(1788,990,20)	(1834,944,40)
# of correct (τ)	234	300	312	329
Time	0/261.1/53.4	0.3/69.3/112.3	0.4/7.6/106.3	0.4/1.7/117.9
Total time	314.5	181.9	114.3	120.0

Table 5 Multilevel algorithm results with different number of levels to create 60 clusters. Timings (coarsening, partitioning and decoarsening) are measured in seconds.

# of level	0	1	2	3	# of clusters
Dm	(1217,1081,15)	(1218,1080,32)	(1306,992,19)	(1330,968,31)	40
Ec	(439,296,33)	(471,264,15)	(498,237,11)	(508,227,16)	30
Hp	(361,455,9)	(369,447,11)	(407,409,16)	(411,405,10)	40
Hs	(420,126,15)	(418,128,14)	(418,128,15)	(426,120,15)	30

Table 6 Multilevel algorithm results for 4 other organisms in DIP: *Drosophila melanogaster* (Dm), *Escherichia coli* (Ec), *Helicobacter pylori* (Hp) and *Homo sapiens* (Hs). The last column has the numbers of clusters used for this experiment.

starts increasing. Particularly the time for partitioning clearly decreases although the time for coarsening increases with the number of levels. The refining step takes up most of time for ML with 3 levels: 117.9 out of 120 seconds. This is because KL refinement algorithm has $O(N^2)$ complexity. So we can improve the timings even further using more efficient refining algorithms such as the Fiduccia–Mattheyses linear time heuristic [9].

Figure 2 and *Table 5* show that the biggest improvement shows at level 1 and the quality of clustering is mostly increasing slowly after that. The optimum number of levels is expected to vary depending on various properties of a network. So for the moment, the number of levels can be decided empirically. There is one weakness of the ML algorithm: we do not know how many levels of coarsening should be used, especially when coarsening includes a random algorithm. But even with one level of coarsening we can expect almost 80% of the best case. Usually we see at least 20% improvement by applying the ML idea.

We also present the computational results for 4 other organisms of DIP in *Table 6* (we exclude *Caenorhabditis elegans* and *Musmusculus* because the residual networks are too small to apply the ML algorithm). Since they all have different numbers of proteins and interactions, we use different number of clusters. Our experiments show that the ML algorithm improves the clustering results in all but one case tested (*Homo sapiens*, Hs).

7 Conclusion

This research focuses on matching groups of proteins which are more likely to be part of the same functional modules. These groups are considered as single nodes so the graph with them is much smaller than the original graph. We presented greedy-based ML algorithms for unweighted graphs which represent protein-protein interactions and compared their computational results. Our algorithm does not lead to the smallest coarser graphs possible but merges more correct vertices. The notion of diagonal-edge weighting is proposed and proved to improve the ML algorithms.

One weakness of ML algorithm is that we do not know how many levels of coarsening should be taken. A tool may help decide the number of levels is the triplet κ as in *Table 5*. Even though the variables τ and κ do not behave identically, κ can be used as a good indicator for picking the number of levels.

This research concludes that ML approaches achieve not only less computation time but also more accurate groupings of proteins for spectral clustering.

Further applications of our clustering algorithm may include other complex networks such as genetic networks, the World Wide Web, citation networks, biological networks and social networks [17].

Acknowledgments

We thank Dr. Alex Pothen for email correspondence about his work on a yeast protein interaction network and reviewers for pointing out minor mistakes and giving sincere advices. A preliminary version of some results presented in this paper will appear in the proceedings of the International Workshop in Bioinformatics Research and Applications (IWBRA 2006). This work was supported partially by the NSF GRANT 0213305.

References and Notes

1 Abou-Rjeili, A. and Karypis, G. (2005) 'Multilevel Algorithms for Partitioning Power-Law Graphs', *Technical Report TR 05-034*, University of Minnesota.

2 Bader, G.D. and Hogue, C.W. (2003) 'An automated method for finding molecular complexes in large protein interaction networks', *BMC Bioinformatics*, Vol. 57, No. 1. Available via `http://www.biomedcentral.com/bmcbioinformatics/archive/` .

3 Batagelj, V. and Zaversnik, M. (2002) 'Generalized Cores', *Computing Research Repository (CoRR)*, Vol. cs.DS/0202039. Available via `http://arxiv.org/abs/cs.DS/0202039` .

4 Bornholdt, S. and Schuster, H.G. (eds.) (2003), *Handbook of Graphs and Networks*, Wiley VCH.

5 Deane, C.M., Salwinski, L., Xenarios, I. and Eisenberg, D. (2002) 'Protein interactions: two methods for assessment of the reliability of high throughput observations', *Mol. Cell Proteomics*, Vol. 1, No. 5, pp. 349–56.

6 Dhillon, I., Guan, Y. and Kulis, B. (2005) 'A fast kernel-based multilevel algorithm for graph clustering', *Proc. The eleventh ACM SIGKDD international conference on Knowledge discovery in data mining.*

7 Ding, C., He, X., Meraz, R. and Holbrook, S. (2004) 'A Unified Representation for Multi-Protein Complex Data for Modeling Protein Interaction Networks', *Proteins: Structure, Function, and Bioinformatics*, Vol. 57, pp. 99–108.

8 Ding, C., He, X., Zha, H., Gu, M. and Simon, H. (2003) 'A MinMaxCut spectral method for data clustering and graph partitioning', *Technical Report TR 54111*, Lawrence–Berkeley National Laboratory.

9 Fiduccia, C.M. and Mattheyses, R.M. (1982) 'A linear time heuristic for improving network partitions', *Proc. 19th IEEE Design Automation Conference*, pp. 175–181.

10 Gabow, H.N. (1990) 'Data Structures for Weighted Matching and Nearest Common Ancestors with Linking', *Proc. SIAM/ACM Symposium on Discrete Algorithms (SODA)*, pp. 434–443.

11 Hagen, L. and Kahng, A. (1991) 'Fast spectral methods for ratio cut partitioning and clustering', *Proc. IEEE Internat. Conf. Computer Aided Design*, pp. 10–13.

12 Karypis, G. and Kumar, V. (1995) 'MeTis: Unstructured Graph Partitioning and Sparse Matrix Ordering System and Version 2.0'.

13 Karypis, G. and Kumar, V. (1998) 'A fast and high quality multilevel scheme for partitioning irregular graphs', *SIAM J. Sci. Comput.*, Vol. 20, No. 1, pp. 359–392.

14 Kernighan, B.W. and Lin, S. (1970) 'An efficient heuristic procedure for partitioning graphs', *The Bell System Technical Journal*.

15 Krogan, N. et al. (2006) 'Global landscape of protein complexes in the yeast Saccharomyces cerevisiae', *Nature*, Vol. 440, No. 7084, pp. 637–43.

16 Monien, B., Preis, R. and Diekmann, R. (2000) 'Quality matching and local improvement for multilevel graph-partitioning', *Parallel Comput.*, Vol. 26, pp. 1609–1634.

17 Newman, M.E.J. (2003) 'Properties of highly clustered networks', *Physics Review*, Vol. 57, pp. 99-108.

18 Oliveira, S., Seok, S.C. (2005) 'A Multi-level Approach for Document clustering', *Proc. ICCS 2005, Lecture Notes in Computer Science*, Vol. 3514, pp. 204–211.

19 Ramadan, E., Osgood, C. and Pothen, A. (2005) 'The Architecture of a Proteomic Network in the Yeast', *Proceedings of CompLife2005, Lecture Notes in Bioinformatics*, Vol. 3695, pp. 265–276.

20 Shi, J. and Malik, J. (2000) 'Normalized cuts and image segmentation', *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 22, No. 8, pp. 888–905.

21 Vazirani, V. (1994) 'A Theory of Alternating Paths and Blossoms for Proving Correctness of the $\sqrt{(}E)$ General Graph Maximum Matching Algorithm', *Combinatorica*, Vol. 14, pp. 71–109.

22 Xiong, H., He, X., Ding, C., Zhang, Y., Kumar, V. and Holbrook, S. (2005) 'Identification of Functional Modules in Protein Complexes via Hyperclique Pattern Discovery', *Proc. Pacific Symposium on Biocomputing (PSB)*, Vol. 10, pp. 221–232.

23 Zhang, L., Wong, S., King, O. and Roth, R. (2004) 'Predicting co-complexed protein pairs using genomic and proteomic data integration'. *BMC Bioinformatics*, Vol. 5, No. 38. Available via `http://www.biomedcentral.com/bmcbioinformatics/archive/` .