A Multilevel Approach to Identify Functional Modules in a Yeast Protein-Protein Interaction Network

S. Oliveira and S. C. Seok

Department of Computer Science, 14 MLH,
University of Iowa, Iowa City IA 52242, USA
{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 [9, 4, 10, 6]. A spectral clustering method plays a critical role identifying functional modules in a yeast protein-protein network in [6, 4]. We present an unweighted-graph version of a multilevel spectral algorithm which more accurately identifies protein complexes with less computational time.

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 topic in computational systems biology. There has been a lot of recent computational approaches to disclose the underlying biological structures [9, 4, 10, 6].

Pothen et al. [6] propose a two-level architecture for a yeast proteomic network. They construct a smaller network from a protein-protein interaction network by removing proteins which interact with too many or too few proteins. A clustering algorithm is applied to this residual network. Validation of clusters is performed by comparing the clustering result with a protein complex database, called MIPS. A spectral clustering method plays a critical role for identifying functional modules in the protein-protein network in their research.

We successfully applied a multilevel spectral algorithm to cluster a group of documents in [16] using similarity matrices which are mostly dense with entries between 0 and 1. Like large-scale networks, the vertex connectivities of proteomic networks follow a scale-free power-law distribution. That is, the proteomic network consists of a small number of high degree nodes and a majority of low degree nodes. However, there are no edge weights. In this paper, we present an unweighted-graph version of a multilevel spectral algorithm in [16] which identifies more protein complexes with less computation time.

Multilevel algorithms have a long history, mostly for PDE in numerical analysis but also for graph partitioning such as in METIS [12]. Recently multilevel schemes have been applied to graph clustering [16, 11]. Multilevel algorithms conventionally consists 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. Multilevel algorithms not only improve qualities of the clustering but also significantly reduce computational time.

The most well-known algorithms include *random-edge matching* (REM) and *heavy-edge* matching (HEM) [13]. And there are more recent matching algorithms like LAM [1]. These algorithms are mainly designed for weighted graphs. We also compared two different coarsening algorithms for weighted graphs in [16]. We have shown that when nodes are merged in order from the heaviest edge weight we can expect better results. We call this Sorted Matching (SM), because each edge weight represents how close two nodes are. The unweighted or uniform weight graph cases can not use any of these edge-oriented methods directly because all edge weights are initially all 1. But after one level of coarsening some edges may represent more than one edge (up to 4). That is, we have groups of different edge weights after coarsening when we define the weight of an edge as the sum of edge weights combined into it. So after one level of coarsening we are able to use Sorted Matching (SM) for unweighted graph coarsening: merge nodes with highest edge weight. But there are many nodes with the same edge weight. Thus we give the higher priority to the edge with fewer combining node weights, which is defined as the number of all nodes included in the supernode. The maximum edge with $1/w(n_i) + 1/w(n_j)$ is taken as a tie-break rule, where $w(n_i)$ and $w(n_j)$ are the node weights (the number of nodes) of supernodes n_i and n_j. Let us call this algorithm Heavy-Edge-Small-Node (HESN). We show that HESN algorithm outperforms a randomly matching algorithm and a matching algorithm which focuses on maximizing the number of nodes collapsed.

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

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 [15].

2 Background on Multilevel Approach and Proteomic Networks

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 representation is very useful to store weights for edges and vertices. We can also use a lot of 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.

2.1 Multilevel Algorithms

The basic concept of multilevel clustering algorithms is that when we have a big graph $G = (V, E)$ to partition, we construct a smaller graph $\bar{G} = (\bar{V}, \bar{E})$ 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 reverse process is called decoarsening.

Coarsening and decoarsening steps are implemented by multiplying a special coarsening matrix C by the graph matrix S. Each column of C has 1's for vertices to merge and 0's for the rest. The way merging or matching is accomplished is explained in section 3.1. Then a series of $S_0, S_1, \cdots, S_l$ are recursively constructed using $C_1, \cdots, C_l$ in the form of $S_i = C_i' * S_{i-1} * C_i$ with $i = 1, \cdots, l$. The coarsest matrix is used to get the initial partition Cut.

Partitioning algorithms fall into two categories: direct partitioning and recursive bipartitioning. One recursive bipartitioning algorithm which has been successfully applied to identify functional modules is Divisive MinMaxCut algorithm [4, 6]. Divisive MinMaxCut algorithm repeatedly performs two main steps. One is selecting a cluster to split and the other is applying the two-way MinMaxCut algorithm. The two-way MinMaxCut algorithms tries to find a pair of disjoint subsets (A, B) of V which minimizes the objective function

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

where $s(A, B) = \sum_{i \in A, j \in B} s_{ij}$. It is well known [3] that the optimal solution of (1) is the eigenvector q_2 associated with the second smallest eigenvalue of the system $(D - S)q = \lambda Dq$, where $D = diag(d_1, d_2, \cdots, d_n)$ and $d_i = \sum_j s_{ij}$.

The optimum value of (1) is called the cohesion of the cluster and is an indicator which shows how closely vertices of the current cluster are related [3].

Divisive MinMaxCut algorithm recursively chooses a cluster which has the least cohesion value, until we have a predefined number of clusters or when all current clusters have cohesion values greater than a threshold.

Decoarsening is the way back to the original graph. The partition from the coarsest level is mapped into finer levels by multiplying a proper coarsening matrix C. Then a Kernighan-Lin (KL) type refinement algorithm [14] is applied to improve the quality at each level. KL starts with an initial partition; it iteratively searches for nodes from each cluster of the graph if swapping of a node to one of the other clusters leads to a better partition. For each node, there would 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 Functional Modules in Protein-Protein Interaction Networks

Proteomic networks have two important features [2]. One is that the degree distribution function $P(k)$ follows a power law (and so is considered a scale-free network). That is, most vertices have low degrees and a few are highly connected. 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 size.

DIP is a protein interaction database. The protein-protein interactions can be detected by high-throughput experiments. However, many of them (as high as 50%) are false positive. Database of Interacting Proteins (DIP: http://dip.doe-mbi.ucla.edu) provides a protein-protein interaction dataset in the budding yeast, Saccharomyces cerevisiae, that is experimentally determined and cross-referenced to the major sequence databases (SWISS-PROT, GenBank, PIR). 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 Deane at al.[5].

Pothen et al. presented a two-level architecture on this CORE dataset. 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 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 17471 edges. We construct a residual network after removing nodes that have degree 20 or more and 3 or less from the original DIP dataset. We do this in similar way to Pothen et al. [6]. The residual network has 1078 nodes and 2778 edges.

MIPS is the Munich information center for protein sequence information on 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 id known proteins and 2429 id unknown proteins. 800 proteins are common to the residual network of DIP protein-protein network.

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 bigger, complex, Tubulin-associated motorproteins. Similarly, all proteins in complexes whose id start with 140 are included in highest complex Cytoskeleton (id:140). Let us call these smallest functional units leaf modules.

3 Coarsening Unweighted Graphs and Computational Experiments

We first present three different matching algorithms for unweighted graphs. A computational comparison of them follows. And then we show how multilevel approach improves the current clustering algorithms. We use two measures to validate our computational results. One is the triplet, κ. The first item of κ is the total number of edges inside clusters and the second is number of edges between clusters. The last item is the maximum number of edges between any two clusters. This measure does not exactly show how well our network is clustered. However, this measure provides a some insight. The bigger 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 exactly clustered. Each cluster partitioned from the DIP residual network is compared with all leaf functional modules of MIPS. We define τ as the sum of 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.

3.1 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 [17] and some aim to maximize the sum of all edge weights [8]. These algorithms are too time intensive [1] and designed for weighted graphs. 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 (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 (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 in a matching step. Similarly, a node weight is defined as the total number of nodes merged into it. We developed a matching algorithm which uses the edge weights and the node weights. After one level of coarsening with RVRM or LVRM, some edges may represent more than one edge (up to 4). That is, we have groups of different edge weights after coarsening. So we now can adapt the Sorted Matching (SM) idea for the unweighted graph coarsening: merge nodes with highest edge weight. But there are too many nodes with the same edge weight. Then we give the higher priority to the edge with lower combined node weights, like LVRM. 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 Heavy-Edge-Small-Node (HESN).

3.2 Comparison of Coarsening Algorithms

Table 1 shows the comparison of three coarsening algorithms without partitioning and refinement. The total numbers of correctly matched proteins are listed after various levels of coarsening. The first row for each algorithm has the sizes of the graphs (no. of nodes) at the various levels of our multilevel scheme. LVRM results in the smallest graph in coarsest level than other two algorithms. But HESN generates more cohesive clusters as the second row of each coarseningon 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.

Table 1. Comparison of different coarsening algorithms with different levels. First row of each algorithm has sizes of graphs. Second rows have the number edges inside, between and maximum between edges (κ) . Third rows have the number correctly grouped nodes (τ).

level	3	4	5	
RVRM	607-340-185	612-346-192-105	616-339-184-100-53	size of graph
	1036-1742-17	1216-1562-22	1283-1495-41	κ
	459	346	218	τ
LVRM	573-296-151	573-302-157-79	571-302-156-79-40	size of graph
	1062-1716-18	1226-1552-17	1254-1524-23	κ
	419	282	182	τ
HESN	601-333-182	601-333-182-102	601-333-182-102-56	size of graph
	1351-1427-37	1602-1176-11	1749-1029-8	κ
	514	397	295	τ

3.3 Various Levels and Identifying Functional Modules

One issue when we use this spectral clustering algorithm is that (1) may have 0 for denominators because all diagonal entries of S are 0. Then the objective function values can be computed without including the clusters whose inner similarity $s(A, A)$ is 0. We tried to add weights on diagonal entries of S with 1's and with degrees. To compare these three algorithms, we generated two groups of clusters without multilevel algorithm. Table 2 shows that we expect better results when we add degrees of nodes into diagonal entries.

Table 3 shows the effect our multilevel algorithm (ML) on finding functional modules. We considered the ML algorithms with levels 0 through 3 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 third row shows the timings for the 3 parts

Table 2. 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.

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 3. Multilevel algorithm results with different number of levels. Timings are measured in seconds.

level	0	1	2	3
no. of nodes	601	333	182	102
edge info (κ)	1490 1288 46	1769 1009 31	1788 990 20	1834 944 40
no. 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

of ML (coarsening, partitioning and decoarsening). The total time consumed decreases up to 3 levels and then 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 takes $O(N^2)$ complexity. So we can improve the timings even further using more efficient refining algorithms such as Fiduccia-Mattheyses linear time heuristic [7].

4 Conclusion

We presented a multilevel algorithm for unweighted graphs which represent protein-protein interactions. 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 showed that this multilevel approach favors not only less computation time but also more accurate groupings of proteins.

References

1. R. Diekmann B. Monien, R. Preis. Quality matching and local improvement for multilevel graph-partitioning. *Parallel Comput.*, 26(12):1609–1634, 2000.
2. S. Bornholdt and H.G. Schuster, editors. *Handbook of Graphs and Networks*. Wiley VCH, 2003.
3. R. Meraz S. Holbrook C. Ding, X. He. A unified representation for multi-protein complex data for modeling protein interaction networks. *Proteins: Structure, Function, and Bioinformatics*, 57:99–108, 2004.

4. Richard F. Meraz Stephen R. Holbrook Chris Ding, Xiaofeng He. A unified representation of multiprotein complex data for modeling interaction networks. *Proteins: Structure, Function, and Bioinformatics*, 57(1):99–108, 2004.
5. Xenarios I Eisenberg D. Deane CM, Salwinski L. Protein interactions: two methods for assessment of the reliability of high throughput observations. *Mol Cell Proteomics.*, 1(5):349–56, May 2002.
6. A.Pothen E.Ramadan, C.Osgood. The architecture of a proteomic network in the yeast. International Workshop on Distributed Data Mining in Life Science (LifeDDM05), 2005.
7. C.M. Fiduccia and R.M. Mattheyses. A linear time heuristic for improving network partitions. pages 175–181. 19th IEEE Design Automation Conference, 1982.
8. H. N. Gabow. Data structures for weighted matching and nearest common ancestors with linking. In *SODA*, pages 434–443, 1990.
9. C.W. Hogue G.D. Bader. An automated method for finding molecular complexes in large protein interaction networks. *BMC Bioinformatics*, 4(1), 2003.
10. C. Ding Y. Zhang V. Kumar S. Holbrook H. Xiong, X. He. Identification of functional modules in protein complexes via hyperclique pattern discovery. Pacific Symposium on Biocomputing (PSB 2005).
11. B. Kulis I. Dhillon, Y. Guan. A fast kernel-based multilevel algorithm for graph clustering. The eleventh ACM SIGKDD international conference on Knowledge discovery in data mining, 2005.
12. G. Karypis and V. Kumar. *MeTis: Unstrctured Graph Partitioning and Sparse Matrix Ordering System, Version 2.0*, 1995.
13. G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. Technical Report 95-035, Department of Computer Science, University of Minnesota, Minneapolis, MN, 1998.
14. B.W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *The Bell System Technical Journal*, 1970.
15. M. E. J. Newman. Properties of highly clustered networks. *Physics Review*, 57:99–108, 2003.
16. S. Oliveira and S.C. Seok. A multi-level approach for document clustering. *Lecture Notes in Computer Science*, 3514:204–211, Jan 2005.
17. V. Vazirani. A theory of alternating paths and blossoms for proving correctness of the (sqrt{E}) general graph maximum matching algorithm. *Combinatorica*, 14(1):71–109, 1994.