Triangular Clique Based Multilevel Approaches to Identify Protein Functional Modules

S. Oliveira[1] and S. C. Seok[1]

Department of Computer Science, 14 MLH, University of Iowa, Iowa City IA 52242, USA
Email:{oliveira, sseok}@cs.uiowa.edu
Phone: (319)-335-0731, (319)-353-4851
Fax: (319)-335-3624

Abstract. Identifying functional modules is believed to reveal most cellular processes. There have been many computational approaches to investigate the underlying biological structures [1, 5, 10, 13]. A spectral clustering method plays a critical role identifying functional modules in a yeast protein-protein network in [10]. One of major obstacles clustering algorithms face and deal with is the limited information on how close two proteins with or without interactions are. We present an unweighted-graph version of a multilevel spectral algorithm which identifies more protein complexes with less computational time [9]. Existing multilevel approaches are hampered with no preliminary knowledge how many levels should be used to expect the best or near best results. While existing matching based multilevel algorithms try to merge pairs of nodes, we here present a new multilevel algorithms which merges groups of three nodes in triangular cliques. These new algorithms produce as good clustering results as previously best known matching based coarsening algorithms. Moreover, our algorithms use only one or two levels of coarsening, so we can avoid a major weakness of matching based algorithms.

Topic: Computing in Biosciences, Data Processing, Numerical Methods

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 has been many recent computational approaches to disclose the underlying biological structures.[1, 5, 10, 13]

Successful approaches to clustering functional modules include partition-based algorithms [5, 10]. Pothen et al. proposed a two-level architecture for a yeast proteomic network [10] and Ding et al. introduced a partitioning algorithm on a bipartite model. Pothen et al. construct a smaller network from a protein-protein interaction network by removing proteins which interact with too

many or too few proteins. And then a spectral clustering method was applied to identify functional modules in the protein-protein network in their research.

The biggest obstacle to identifying functional modules is that protein-protein interaction networks are unweighted or uniformly weighted. Unweighted graphs are considered to be harder to partition than weighted graphs because unweighted graphs provide only limited information on the strength of connection between two vertices. Multilevel (ML) algorithms have been introduced to identify more functional modules based on matching algorithms in PPI networks [9]. But these algorithms are hampered by two weaknesses. First, it is hard to find the optimal number of levels. Second, most matching based algorithms use random algorithms, so the clustering results vary from an experiment to another.

Our Triangular Clique (TC) based multilevel algorithm was inspired by Spirin et al's approach to investigate the large-scale structure of PPI networks [12]. They used the Clique idea to identify highly connected clusters of proteins in protein-protein interaction networks. They not only enumerated all cliques of size 3 and larger (complete subgraphs) but also partially complete subgraphs with high quality.

Our algorithm use only triangular cliques (cliques of size 3). These are different from matching based ML algorithms which pick pairs of nodes based on edge or node related information to merge, TC based algorithms try to merge highly connected triples of nodes. We present four different kinds of TC based algorithms according to the decision on how to deal with two TCs which share one or two nodes. These algorithms are compared and analyzed with the computational results.

We show some TC based algorithms identify as good as or better functional modules with one or two levels of coarsening than the best matching based ML algorithm we found in [9].

2 Features of Interaction networks and Two-level Approach

Graph theory is commonly used as a method for analyzing protein-protein interaction (PPI) networks in Computational Biology. Each vertex represents a protein, and edges correspond to experimentally identified PPIs. Proteomic networks have two important features [3]. One is that the degree distribution function $P(k)$ follows a power law (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. 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.

A two level approach was proposed by Pothen et al. [10] to identify functional modules in a proteomic network in yeast. The main idea is derived from the k-cores concept which was introduced by Batagelj, Mrvar and Zaversnik [2] in graph theory. 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.

3 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 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. Our ML algorithms use this matrix representation.

3.1 Multilevel Spectral 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. A series of matrices $S_0, S_1, \cdots, S_l$ are recursively constructed using $C_1, \cdots, C_l$ in the form of $S_i = C_i' * S_{i-1} * C_i$ 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 is done by using a recursive spectral bipartitioning (divisive partitioning). 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.

The best known spectral clustering algorithms is the MinMaxCut algorithm [6]. Two-way MinMaxCut clustering algorithm aims to minimize

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

where $s(A,B) = \sum_{i \in A, j \in B} s_{ij}$.

In [6], a continuous approximation to this problem has the solution which is the eigenvector q_2 associated with the second smallest eigenvalue of the system $(D-S)q = \lambda Dq$, 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 $B = \{q_2(i) \mid q_2(i) \geq q_2(opt)\}$.

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 [6]. This value can be used for the cluster selection algorithm. Divisive algorithms recursively choose a cluster which has the least cohesion for partitioning 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 [7]. KL starts with an initial partition; it iteratively searches for nodes from each cluster of the graph if 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.

3.2 Matching based Coarsening Algorithms

A matching in a graph is a set of edges in which no two of them are incident on the same node. We introduced a heuristic matching algorithm which works very well on weighted graphs in [8], called Sorted Matching (SM). SM was used earlier by us to improve clustering results for groups of documents, which compose weighted graphs. In SM, nodes are merged in order of decreasing edge weight.

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.

Our matching algorithm for unweighted graphs introduced in [9] goes as follows: 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 a matching step. A node weight is defined as the total number of nodes merged into it.

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 then give the 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).

HESN was introduced and shown to outperform the other matching based algorithms [9].

4 Coarsening with Triangular Cliques

Matching based coarsening algorithms merge groups of at most two nodes which have an edge between them. These algorithm have worked well especially on weighted graphs because all edges have different weights. These weights play a key role for picking pairs of nodes to merge. Meanwhile unweighted graphs do not provide this information. The ML algorithm with HESN works well on the unweighted graphs even though HESN is a matching based coarsening. However, in general, matching based ML algorithms have two main weaknesses. First, it is hard to find the optimal number of levels which generates the best clustering. Second, most matching based algorithms has a random component, so the clustering results vary from an experiment to another.

A clique in an undirected graph $G = (V, E)$ is a subset $V' \subseteq V$, where each pair of vertices is connected by an edge. The problem of finding the maximal size of a clique for a given graph is an NP-complete problem [11]. However, finding all cliques comprised of three vertices takes $O(|E|^2/|V|)$ time. We pay attention to that, first, if both proteins p_1 and p_2 interact with the protein p_3, p_1 and p_2 should interact and second, when three proteins p_1, p_2, p_3 forms a triangular clique (TC), the chance all three proteins are clustered in the same functional module is high. We show the quality of triangular cliques in section 5. These three nodes p_1, p_2, p_3 in a graph compose a triangular clique.

When we use TCs to form sets of vertices to be merged, we have to make two decisions. The first one is mainly because many TCs shares one or two nodes

with others. All two pairs of TCs fall into three different cases: sharing no, one, or two common vertices. The first one does not make any difference. We focus on the last two cases. How do we merge TCs for each case? The graph at the upper-left of Figure 1 has four TCs, from TC1 through TC4. TC1 and TC2 share two common nodes and TC3 and TC4 share one common node. One way is to merge all nodes in TCs which share one or two nodes into one supernode, let us call this method TC_ALL. In this case, any group of TCs which share one or more vertices is merged into a supernode. Another simple way is to merge one of two TCs for both cases and leave other vertices unmerged, let us call this TC_ONLY. When we assume that TC1 and TC3 are chosen over TC2 and TC4 and merged into two separate supernodes by TC_ONLY, two nodes of TC4 and one node of TC2 are left unmerged. We consider two variants of TC_ONLY according to how to deal with these three unmerged nodes. The one unmerged node of TC2 forms an edge with the supernode after coarsening and looks reasonable to be grouped in the same cluster with the supernode made of TC1. So we devise a new algorithm, TC_ONE, which is basically the same as TC_ONLY but merging the left out node of TC2 and TC1 into a supernode. Similarly, the two unmerged nodes of TC4 form a TC with the supernode created. The two unmerged nodes have high chance to be merged with newly created supernode into a supernode if we have additional levels of coarsening. If we do not want to merge them with the supernode, we can have all three nodes of TC3 and two nodes of TC4 form two separate supernodes. Let us call this algorithm TC_TWO. In this case, the one unmerged node of TC2 by TC_ONLY is still left unmerged as shown in Figure 1.

Second problem we face is whether or not we keep creating more levels. If we use more than one level then the question is how many levels to create. In this paper we focus on using the same algorithm to create more levels. The performance of these four algorithms is presented and compared with a matching based coarsening algorithm, HESN, in the following section.

5 Model Networks and Computational Experiments

The budding yeast, *Saccharomyces cerevisiae*, is considered the simplest and so is the most investigated organism. Pothen's two-level architecture is applied to the CORE dataset of DIP (Database of Interacting Proteins, dip.doe-mbi.ucla.edu), where the pairs of interacting proteins identified were validated according to the criteria described in Deane et al. [4]. The network has 2610 proteins and 6236 interactions. Their idea is that removing high degree proteins (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 with removing nodes that have degree 20 or

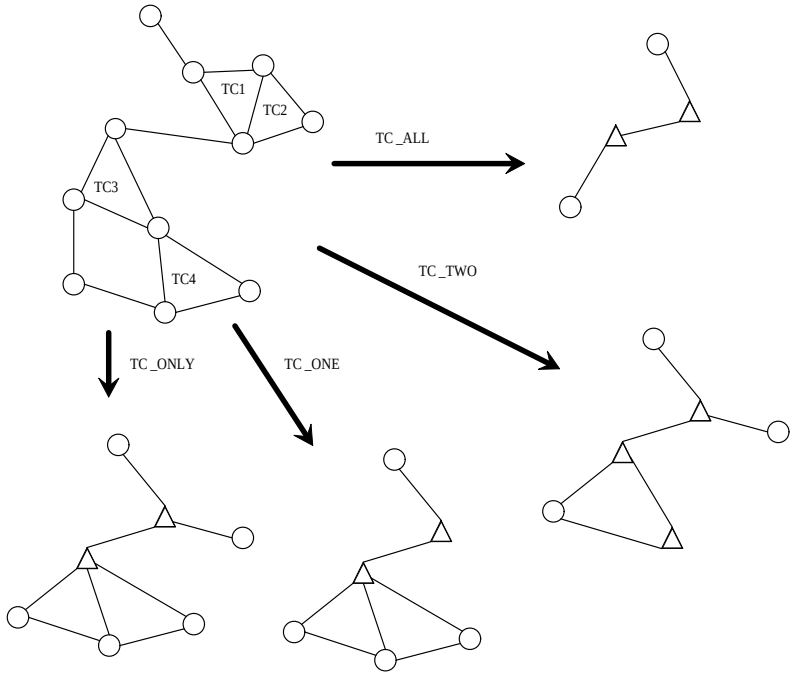

Fig. 1. Four different TC based coarsening algorithms. Each circle stands for a node and a triangle represents a supernode after coarsening.

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.

After our ML spectral algorithm is applied to this residual network, the clustering results are compared with the MIPS (mips.gsf.de) dataset as we did in [9]. Note that the residual network and MIPS dataset share 800 proteins. So the maximum number of correctly clustered nodes is 800 for any experiment.

Now we present various computational results to investigate the properties of TC based ML algorithms. Table 1 shows the number of nodes, the number of TCs, and the number of correctly grouped nodes as the number of levels increases. The sum of the maximum number of proteins which belong to the same functional module in each supernode is what we use as the number of correctly grouped proteins. Notice that, as the number of levels increases, the number of correctly grouped proteins should not increase. The original number of proteins in the residual network is 1078. As expected, TC_ALL collapses the largest number of nodes at each level of coarsening (see the first entries in row TC_ALL). But the quality of grouping worsens pretty fast (see the third entries in row TC_ALL). So with TC_ALL as the number of level increases the size of network shrinks very fast and the quality of coarsening is becoming very

bad. Meanwhile, TC_TWO merges the least after the first level, 819 proteins. TC_ONLY merges more nodes than TC_TWO at the beginning but the least after all. Even though TC_ONLY merges the least nodes after all, the quality of grouping remains good. TC_ONE merges the most nodes except when compared to TC_ALL and the quality of grouping also decreases faster than TC_ONLY and TC_TWO.

There are 1195 TCs found in the original graph. The number of TCs decreases at the first level to as few as 145 for TC_ONE or as many as 580 for TC_TWO. And then the number of TCs does not change much for the first a few levels except TC_ALL for which the number of TCs significantly continues to decrease.

level	1	2	3	4	5
HESN	601	333	182	102	56
TC_ALL	513/272/586	303/198/265	89/39/91	35/4/47	
TC_ONE	662/145/733	550/145/647	501/342/544	468/333/460	438/316/405
TC_TWO	819/580/759	686/261/719	561/147/679	533/133/650	517/140/610
TC_ONLY	770/273/754	630/168/693	578/214/624	554/308/567	540/334/523

Table 1. The comparison of four different TC based coarsening algorithms. Note that $xx/yy/zz$ means xx nodes, yy TCs, and zz correctly grouped nodes.

Finally, we see the clustering results when the ML spectral clustering algorithm is actually applied with the coarsening algorithms. Tables 2 shows the number of correctly clustered proteins with four different TC based coarsening algorithms when 1 up to 5 levels are used to form 40, 60, and 80 clusters. These results are compared with the HESN matching [9] based ML algorithm. Some algorithms, like HESN and TC_ALL, do not have big enough networks to form particular number of clusters after particular levels of coarsening. For example, there are only 56 nodes after three levels of coaresening with HESN, so we do not try to construct 60 and 80 clusters. First row has the results with HESN which is considered to work best among the matching based algorithms in [9]. Without any ML algorithm 201, 234, and 286 proteins are reported to be correctly clustered to form 40, 60, and 80 clusters respectively.

The most remarkable point from the table is the clustering results of TC_ONE and TC_ONLY with one level of coarsening are the best or almost best compared with more than one level of coarsening. As for TC_TWO, the results are improving up to some point. The quality of grouping by TC_TWO is shown well in Table 1. However, the clustering results of TC_TWO are not good compared to TC_ONLY and TC_ONE when these algorithms are actually applied to ML spectral algorithm. While TC_ONLY provides the best results with one level of coarsening, TC_ONE generates the best results with one or two levels. And the quality of clustering of TC_ONLY drops significantly with two levels. We guess that more than one level of coarsening causes overmerging, that is, the quality of grouping with two or more levels is not good enough to improve clustering.

	40	60	80
HESN	232/248/258/253/**267**	300/312/**329**/308	354/364/342/**367**
TC_ALL	**245**/234/235	**299**/286	**346**/321
TC_ONE	257/**261**/253/246/238	**323**/319/319/307/312	368/**370**/358/359/361
TC_TWO	203/230/249/**253**/248	283/296/300/**314**/314	336/**363**/357/355/360
TC_ONLY	**271**/248/248/243/238	**329**/306/316/298/301	**369**/361/367/354/349

Table 2. The numbers of correctly clustered proteins with four different TC based coarsening algorithms and one matching based algorithm to form 40, 60, and 80 clusters, when 1 up to 5 levels are used. First row has the clustering results with a matching based ML algorithm, HESN. Best results are in bold font

6 Conclusion

In this paper we presented Triangular Clique (TC) based multi level algorithms not only to avoid problems caused by matching based algorithms but also to improve the quality of clustering. Triangular Clique based coarsening algorithms works easily by finding TCs in a given graph and then merging in the nodes to form a supernode in the next level as described in section 4. Among our four TC based algorithms TC_ONLY with one level usually gives the best results. TC_ONE also shows almost the same result as TC_ONLY with one or two levels. Our TC based ML algorithms do not rely as much on random algorithms. We believe our TC based algorithms outperforms matching based ML algorithm because TC based algorithms take advantage of the fundamental structure of unweighted graphs.

References

1. G. D. Bader and C. W. Hogue. An automated method for finding molecular complexes in large protein interaction networks. *BMC Bioinformatics*, 4(1), 2003.
2. V. Batagelj, A. Mrvar, and M. Zaveršnik. Partitioning approach to visualization of large graphs. In *Graph drawing (Štiřín Castle, 1999)*, volume 1731 of *Lecture Notes in Comput. Sci.*, pages 90–97. Springer, Berlin, 1999.
3. S. Bornholdt and H. G. Schuster, editors. *Handbook of Graphs and Networks.* Wiley VCH, 2003.
4. C. M. Deane, L. Salwinski, I. Xenarios, and D. Eisenberg. Protein interactions: two methods for assessment of the reliability of high throughput observations. *Mol Cell Proteomics.*, 1(5):349–56, May 2002.
5. C. Ding, X. He, R. F. Meraz, and S. R. Holbrook. A unified representation of multiprotein complex data for modeling interaction networks. *Proteins: Structure, Function, and Bioinformatics*, 57(1):99–108, 2004.
6. C. Ding, X. He, H. Zha, M. Gu, and H. Simon. A minmaxcut spectral method for data clustering and graph partitioning. Technical Report 54111, LBNL, December 2003.
7. B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *The Bell System Technical Journal*, 1970.

8. S. Oliveira and S. C. Seok. A multi-level approach for document clustering. *Lecture Notes in Computer Science*, 3514:204–211, Jan 2005.
9. S. Oliveira and S. C. Seok. A multilevel approach for identifying functional modules in protein-protein interaction networks. *Proceedings of IWBRA 2006, Lecture Notes in Computer Science*, 3992, 2006. to appear.
10. E. Ramadan, C. Osgood, and A. Pothen. The architecture of a proteomic network in the yeast. *Proceedings of CompLife2005, Lecture Notes in Bioinformatics*, 3695:265–276, 2005.
11. S. Skiena. *The Algorithm Design Manual*. New York:Springer-Verlag, 1998.
12. V. Spirin and L. A. Mirny. Protein complexes and functional modules in molecular networks. *Proc Natl Acad Sci U S A*, 100(21):12123–12128, October 2003.
13. H. Xiong, X. He, C. Ding, Y. Zhang, V. Kumar, and S. Holbrook. Identification of functional modules in protein complexes via hyperclique pattern discovery. In *Pacific Symposium on Biocomputing (PSB 2005)*, volume 10, pages 221–232, 2005. Available via `http://psb.stanford.edu/psb-online/`.