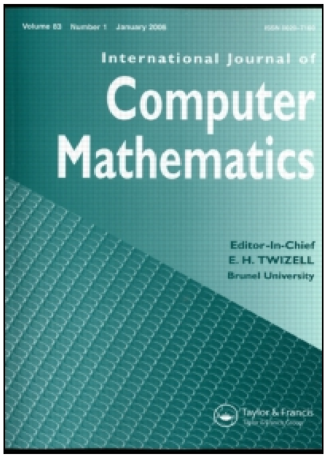

International Journal of Computer Mathematics
Publication details, including instructions for authors and subscription information:
http://www.informaworld.com/smpp/title~content=t713455451

Multilevel approaches for large-scale proteomic networks

S. Oliveira [a]; S.-C. Seok [a]
[a] Department of Computer Science, University of Iowa, Iowa City, Iowa, USA

Online Publication Date: 01 May 2007
To cite this Article: Oliveira, S. and Seok, S.-C. (2007) 'Multilevel approaches for large-scale proteomic networks', International Journal of Computer Mathematics, 84:5, 683 - 695
To link to this article: DOI: 10.1080/00207160701332382
URL: http://dx.doi.org/10.1080/00207160701332382

International Journal of Computer Mathematics
Vol. 84, No. 5, May 2007, 683–695

Multilevel approaches for large-scale proteomic networks

S. OLIVEIRA* and S.-C. SEOK

Department of Computer Science, University of Iowa, Iowa City, Iowa, USA

(Received 30 September 2006; revised version received 08 January 2007; second revision received 06 March 2007; accepted 08 March 2007)

Our multilevel algorithms aim to improve existing graph clustering algorithms which predict protein complexes in large-scale proteomic networks, which are represented as unweighted graphs.

Current matching based multilevel algorithms are hampered by low-quality of grouping (coarsening) even though they dramatically reduce computational time. We present a multilevel algorithm with structured analysis of unweighted networks which constructs high-quality groups of nodes merged before applying a clustering algorithm.

A 2-core network of a proteomic network is constructed by removing all nodes which have degree less than two recursively. Our multilevel algorithm builds a series of smaller (or coarser) networks from the 2-core network by searching highly dense subgraphs in each level and then a clustering algorithm is applied. The clustering results are passed to the original network with additional fine tuning. All leftover nodes outside the 2-core network are added back after the multilevel algorithm.

Compared to existing multilevel algorithm, our multilevel algorithm on 2-core networks shows that nodes in coarser networks have higher accuracy in all supernodes, and clustering results show up to 15% (mostly around 10%) improvements. Moreover, our clustering algorithm uses only one or two levels, so it is free from deciding the number of levels to expect best results.

Keywords: Clique; Multilevel method; Protein complexes; Protein–protein interaction

AMS Subject Classifications: 92D08; 91C20

1. Introduction

Most cellular processes are believed to be carried out by groups of highly interacting proteins called functional modules, protein complexes, or molecular complexes. Recent large-scale high-throughput experiments, and integration of published data, have generated large protein–protein interaction (PPI) networks. In these networks, the nodes represent proteins, and the edges are interacting pairs of proteins. Even the simplest organism, yeast, has more than 5000 proteins. Protein complexes can be detected by identifying highly

*Corresponding author. Email: oliveira@cs.uiowa.edu

International Journal of Computer Mathematics
ISSN 0020-7160 print/ISSN 1029-0265 online © 2007 Taylor & Francis
http://www.tandf.co.uk/journals
DOI: 10.1080/00207160701332382

connected sets of proteins in the PPI networks. Computational identification of functional modules or protein complexes can provide an inexpensive guideline for biological experiments.

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. Moreover, large-scale networks demand excessive computational time as the size increases.

Earlier on, Multilevel (ML) algorithms have been studied in diverse computational settings such as for solving linear systems arising from partial differential equations and graph partitioning. Recently, we have presented multilevel approaches for graph clustering algorithms for PPI networks [1–3]. The basic idea of ML algorithms is to reduce the problem to a smaller one so that sophisticated approaches can be applied at a lower cost. This is similar to the idea of solving a problem at a macroscopic level as opposed to a microscopic level. To represent the larger problem as a smaller model, we need a coarsening method for reducing the PPI network to a smaller network. Combining nodes of the graph in this coarsening process can be accomplished by matching pairs of nodes. Matching algorithms try to merge nodes based on edge or node related information. PPI matching algorithms were presented in [1]. However, matching algorithms on PPI networks are hampered by the lack of closeness information between nodes, i.e. no edge weights on the PPI network. While most multilevel algorithms are matching based, there are some group-based algorithms introduced in [4]. Many of these algorithms fail to merge correct pairs of proteins because they do not take advantage of any structured analysis.

In [2] we developed Triangular Clique (TC) based algorithms which merge highly connected triples of nodes. A *clique* is a set of nodes in a graph where each node in the clique is connected to every other node in the clique by an edge of the graph. A *triangular clique* is a clique of three nodes (that is, three nodes whose edges form a triangle). Our TC-based multilevel algorithm was inspired by Spirin and Mirny's use of cliques to identify highly connected clusters [5]. Our TC based algorithm showed more proteins correctly merged into supernodes than any other matching based algorithm. This is because all three proteins in a TC have a very high chance of being part of the same functional module. A weakness of the TC based algorithms is that there are many TCs in even a moderately-sized clique. For example, there are 560 TCs in a clique of 16 nodes.

Another recent approach for identifying protein complexes used maximal cliques to create subgraphs with high densities [6]. Cliques are called maximal when they are not completely contained in another. In general, enumerating all maximal cliques takes much more time than finding all TCs. Fortunately, PPI networks are quite sparse, so all maximal cliques are enumerated quickly.

Our experimental results show the quality of maximal cliques for finding protein complexes are very high. The bigger the maximal clique, the higher the quality of the results. This strongly motivates us to consider maximal cliques in conjunction with our multilevel algorithms. In this paper we generalize our TC-based multilevel algorithms to include maximal cliques. Note that the maximal cliques themselves do not identify clusters completely because they may have overlaps among them. Instead, they are used to quickly identify highly connected disjoint subgraphs. We call these new approaches structure-based multilevel algorithms.

We apply the new algorithms to a recently reported protein–protein interaction network in the yeast *Saccharomyces cerevisiae* [7]. The number of levels to produce best results is a common issue in multilevel algorithms. TC based algorithms showed that one level of grouping (or coarsening) is enough to produce clustering results as good as other matching based algorithms [2]. Similarly, the new maximal clique based multilevel algorithms achieves this with one or two levels of coarsening.

2. Methods

2.1 *Model PPI network and protein complex dataset*

Proteomic networks have two important features [8]. One is that the degree distribution function $P(k)$ (the number of nodes of degree k) follows a power law $P(k) \approx C\,k^{-\alpha}$ (and so is considered a scale-free network). This means that, most nodes 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 number of nodes.

Protein–protein interaction data have been available from various high-throughput experimental methods [9–13]. There are also many websites which provide PPI data on the Internet such as BIND, DIP, IntAct, MINT and Mpact (MIPS). Tandem affinity purification (TAP) of affinity-tagged proteins followed by mass spectrometry (MS) have accurately identified many protein–protein interactions [14]. Krogan *et al.* recently reported a list of protein complexes along with a group of protein–protein interactions in the yeast *Saccharomyces cerevisiae* [7]. They identified more interactions with better accuracy than the previously introduced TAP-MS method [11]. In their new method, 4562 TAP-tagged proteins go through two independent MS methods in parallel. A machine learning algorithm then integrates the two data sets and assign probabilities to interactions. After generating interactions with probability scores, they defined a 'core' data set of 7123 protein–protein interactions with 2708 proteins with a certain cut-off value on the scores. Let Y represent the network generated, consisting of 2708 nodes and 7123 edges. This is the core data set which is considered in this paper. More interactions with a lower cut-off value consist of an 'extended' set of over 14,000 interactions involving more than 3600 proteins. A Markov clustering algorithm was applied to construct 547 distinct protein complexes. About half of these were not annotated in MIPS or two other data sets (cataloged in Cellzome and BIND respectively), using affinity purification and mass spectrometry method. This newly identified data set outperforms previous protein complex data sets in precision and homogeneity.

In this paper we present computational results for the network of Krogan *et al.* [7]. In [15] we present results of these algorithms applied to the PPI network and the protein complexes cataloged in MIPS. The MIPS Comprehensive Yeast Genome Database (CYGD) provides the catalog of protein–protein interactions, the protein complex catalog and the protein localization catalog which stores information related to the proximity of proteins in *Saccharomyces cerevisiae* [16]. The PPI network was created manually from more than 3000 single experiments and published large-scale experiments. The protein complexes currently include more than 200 manually extracted protein complexes, and 3 systematic analysis of large-scale experiments.

2.2 *On 2-core networks*

One of the distinct features of PPI networks is that they follow a power-law property [8]. These networks have many low-degree nodes and a relatively small number of high-degree nodes. Very high-degree nodes correspond to proteins that interact with most other proteins, and these interactions can obscure the connections between other proteins. The low-degree nodes not only increase time complexity but also makes it harder for clustering algorithms to identify highly connected subgraphs, even though they can be grouped easily compared to high-degree nodes.

The k-core idea was originally suggested as a tool to simplify complex graph structures by Seidman in 1983 [17]. If we repeatedly remove nodes of degree $<k$ from a graph (along

with the edges associated with the removed nodes) we will eventually obtain a graph where all nodes have degree $\geq k$. The graph that remains is called the k-core of a graph. This idea is very useful and well suited to scale-free networks like PPI networks [18, 19]. Nodes which are part of some cycle are called cyclic nodes, and those that are not are called non-cyclic nodes.

In order to clarify the following discussion, we introduce the following notation. Let k-core(H) be the k-core of the graph H. Let component$(H) = \{ v \in V(G) \mid v$ is connected to $H\}$. Let 1-path$(H) = \{v \in V(G) \mid$ there is exactly one path joining v to $H\}$. Also $A\backslash B$ is the set of elements in A that are not in B. If A is a network, then $A\backslash B$ is the network whose nodes are the nodes in A but not in B, with all edges in A that do not contain a node in B.

In Krogan *et al.*'s PPI network with 2708 nodes as described in the previous section, there are 1717 nodes in 2-core(Y), leaving 991 nodes in Y that are not in 2-core(Y). Of these 991 nodes, 852 are connected to 2-core(Y) by a single path. The remaining $991 - 852 = 139$ nodes in $Y\backslash$1-path(2-core(Y)) comprise 59 separate connected components; the largest of these connected components has seven nodes, and 16 of the 59 components have three or more nodes. These components of $Y\backslash$1-path(2-core(Y)) have high overall accuracy (131 out of 139 nodes or 94.2% are grouped with nodes in the same functional module). Each node in 1-path(2-core(Y)) can be assigned to one of the clusters in 2-core(Y).

For each node v in 1-path(2-core(Y))$\backslash$2-core(Y) there is one path from v to 2-core(Y); we have an initial coarsening where for each node w in 2-core(Y), we form the set $S_w = \{ v \in$ 1-path(2-core(Y))$\backslash$2-core$(Y) \mid$ the path joining v to 2-core(Y) ends at $w\}$. There are 432 nodes w in 2-core(Y) where $S_w \neq \emptyset$. Once a given node w in 2-core(Y) is assigned to a cluster, the nodes of S_w can also be assigned to the same cluster. This approach correctly assigns 1111 (87.3%) out of $1284 = 852 + 432$ nodes to clusters. So after our clustering algorithm is applied to 2-core(Y), adding the nodes in 1-path(2-core(Y))$\backslash$2-core(Y) improves the quality of the clustering.

We apply our multilevel algorithm to this downsized 2-core network. The quality of grouping (or coarsening) is described in detail in section 2.4. This approach not only improves the quality of grouping but also saves time enumerating all maximal cliques. Building the 2-core network is very helpful for our methods, since it reduces the size of the networks, without changing any maximal clique, and the quality of the removed subgraphs is very high.

2.3 *Triangular clique based multilevel approach*

Graph theory is commonly used as a method for analysing protein–protein interaction (PPI) networks in computational biology. Each node represents a protein, and edges correspond to experimentally identified PPIs. Let $G = (V, E)$ be a graph with node 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 nodes. 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 nodes and off-diagonal entries s_{ij} represent edge weights. Given interaction data, we typically set the node weights to one and $s_{ij} = 1$ if there is an interaction between proteins i and j, and $s_{ij} = 0$ otherwise. Our multilevel algorithms use this matrix representation.

The basic concept of 'multilevel clustering' algorithms is that when we have a large graph $G = (V, E)$ to cluster, we construct a smaller graph $\bar{G} = (\bar{V}, \bar{E})$ whose nodes (that is, supernodes) are groups of nodes from G. We can apply a clustering method to this smaller graph $\bar{G}$, and transfer the partition to the original graph. This coarsening can be repeated: the smaller graph can also be coarsened, and so on, giving a sequence of graphs

$G_0 = G,\ G_1 = \bar{G},\ G_2 = \bar{\bar{G}},$ etc. This process stops when the size of the coarsest graph is 'small enough'. This idea is very useful because smaller matrices and graphs require much less time to cluster. The process of constructing the smaller matrix is called coarsening, and the process of transferring the partition to the larger graph is called decoarsening.

If the matrix S represents the graph G (so that $s_{ij} \neq 0$ if there is an edge between nodes i and j), then we can represent the coarsening by a matrix C: $c_{ik} = 1$ if node k of $\bar{G}$ contains node i of the original graph G; $c_{ik} = 0$ otherwise.

Let S_j be the matrix for graph G_j. The three main steps of multilevel cluster algorithms are:

(i) **Coarsening** Constructing a series of matrices $S_0, S_1, \ldots, S_l$ recursively using coarsening matrices $C_1, \ldots, C_{l-1}$ in the form of $S_i = C_i^T \cdot S_{i-1} \cdot C_i$ with $i = 1, \ldots, l$. Each column of a coarsening matrix shows the group of nodes merged.
(ii) **Clustering** Forming a clustering for the coarsest graph with a clustering algorithm.
(iii) **Decoarsening with refinement** Constructing new clusterings for the finer levels and refining them.

A graph clustering algorithm, the Minmaxcut method [20], was used for Pothen's two-level architecture model of a proteomic network in Yeast [19] and Ding's bipartite graph modelling of protein complex data [21]. We also used the clustering algorithm to validate our previous multilevel algorithms. Minmaxcut algorithm is a divisive spectral clustering algorithm which 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 the node set V by computing the eigenvector q_2 associated with the second smallest eigenvalue λ_2 of the graph Laplacian L. That is, $Lq_2 = \lambda_2 q_2$, $q_2 \neq 0$. The graph Laplacian L of a weighted graph G is the matrix where the rows and columns correspond to nodes of G, and $l_{ij} = -w_{ij}$ (w_{ij} being the weight of the edge joining nodes i and j) if $i \neq j$, and $l_{ii} = \sum_{j \neq i} w_{ij}$. Please refer to [20] for more details.

Divisive algorithms recursively choose a cluster which satisfies a selection criterion and divides it, until we have the predefined number of clusters or until all current clusters satisfy a certain condition. We suggested a variant of Ding's Minmaxcut algorithm by adopting the 'diagonal weighting' idea for unweighted network [1]. The diagonal entries of network matrix are initially all zero. We showed that when the degree of each node is assigned to the corresponding diagonal entry, Minmaxcut algorithm works better.

The clustering of graph G_i at level i is represented by a matrix Cut_i where $(Cut_i)_{pq} = 1$ if node p belongs to cluster q, and zero otherwise. The partition from the coarsest level is mapped into finer levels by using the coarsening matrix: $Cut_i = C_i^T \cdot Cut_{i-1}$ where i is the level number of the coarser level.

A traditional Kernighan–Lin (KL) type refinement algorithm can be applied to improve the quality at each level [22], which has $O(N^2)$ complexity. 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. However, this scheme has many redundant computations. For a given node u, all clusters are considered as possible candidates to which the node u moves. Moving an edge to a cluster to which it is not already connected will always increase the number of edges between clusters. So we only calculate the improvements of cut with clusters which have edges with the node u. This modified scheme works much faster than traditional KL, especially for the sparse networks with large number of clusters. It should be mentioned that there are possibilities of certain objective functions which may improve when a node u moves to a cluster which does have any edge with u. Note that it would be a good idea to find a faster refinement algorithm for multiway clustering such as the Fiduccia–Mattheyses linear time heuristic which was used for bipartitioning problems [23].

A key part of the ML algorithm is the coarsening of the network; that is, constructing a smaller network preserving essential information of the original network. Matching based algorithms try to merge at most two nodes based on edge or node related information. We have developed various multilevel algorithms to improve existing graph clustering algorithms for PPI networks based on matching. In [1, 3] we presented a ML based matching algorithm which combines a minimal matching idea and a greedy algorithm for weighted graphs. However, these matching based algorithms are hampered by the lack of information on the best number of levels to use. In addition, it fails to merge correct pairs of proteins because they do not take advantage of any structured analysis.

In [2] we developed Triangular Clique (TC) based algorithms which merge highly connected triples of nodes. A clique in an undirected graph $G = (V, E)$ is a subset $V' \subseteq V$, where each pair of nodes in V' is connected by an edge. Our TC-based multilevel algorithm was inspired by Spirin and Mirny's use of cliques to identify highly connected clusters [5]. They used the 'clique' idea to identify highly connected clusters of proteins in PPI networks. They not only enumerated all cliques of size three or larger (complete subgraphs) but also constructed partially complete subgraphs with high density.

The problem of finding the maximum size of a clique for a given graph is an NP-complete problem [24]. However, finding all cliques comprised of three nodes takes $O(|E|^2/|V|)$ time. Cliques which are not completely contained in another are called maximal. A recent algorithm using cliques tries to construct partially complete cliques by merging overlapping maximal cliques [6]. Even though the protein complexes in [6] are predicted as accurately as Spirin and Mirny's method in fairly fast time, the proteins in the predicted protein complexes cover no more than 60% of all proteins in protein complexes annotated in MIPS with just over 70% accuracy.

When we use TCs to form sets of nodes to be merged, we have to make a decision, considering TCs sharing one or two nodes with each other. Our four TC-based algorithms presented in [2] are devised to deal with overlaps of one or two nodes between any two TCs. The graph at the upper-left of figure 1 has five TCs, TC1 through TC5. TC1 and TC2 share two nodes, and TC3 and TC4 share one 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 sharing nodes and leave the other TC nodes unmerged, let us call this TC_ONLY. Notice that in the figure we assumed 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 were left unmerged. The way we deal with the unmerged nodes in TC_ONLY generates two possible new approaches, which we name TC_ONE and TC_TWO. The one unmerged node of TC2 forms an edge with the supernode after coarsening and it seems reasonable to group it in the same cluster from TC1 created in the TC_ONLY case. We name this approach TC_ONE. In other words TC_ONE is basically the same as TC_ONLY but merges the left-out node of TC2 with the TC1 supernode. TC_TWO however merges all unmerged nodes in TCs which share one or two nodes with the chosen TC. Under the same assumption as above TC1 and TC3 are chosen over other TCs, four nodes in TC1 and TC2 are merged and then five unmerged nodes in TC3, TC4 and TC5 are merged into a supernode. Since TC_ONE, TC_ONLY and TC_TWO all use a certain order to choose TCs, the coarsening varys depending on the order to choose TCs to merge.

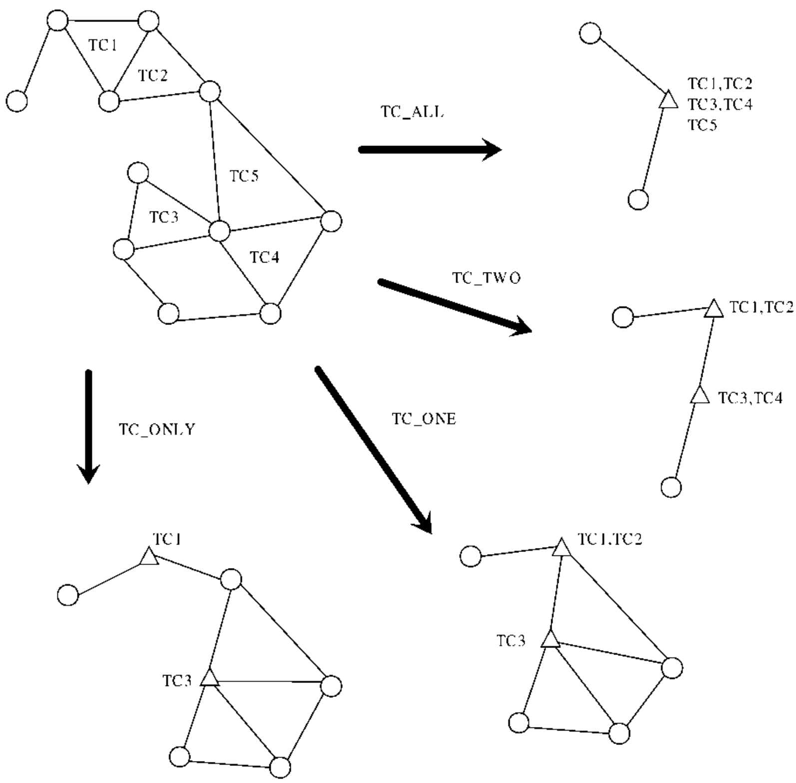

Figure 1. Four different TC based coarsening algorithms. Each circle stands for a node and a triangle represents a supernode after coarsening.

2.4 *Structure based multilevel algorithms with maximal cliques*

The 2-core of Krogan *et al.*'s network has 1717 proteins and can be broken into two groups. One group contains the proteins which belong to a maximal clique. The other group contains the remaining proteins. We deal with each of these groups in a different way.

First we describe how we deal with the nodes in the first group. If we consider cliques of size three or larger, 1143 distinct proteins are found in 1386 maximal cliques, with 5895 appearances. On average, each protein in this group contributes to about 5.2 maximal cliques. The accuracy of a clique is measured by the largest number of proteins from the same protein complex. The accuracy of a group of cliques is the number of nodes that are correctly grouped in all cliques in the group, divided by the number of nodes in the cliques in the group. The accumulated accuracy of a group of cliques is the sum of the number of correctly grouped nodes (summed over all cliques), divided by the sum of the sizes of the cliques. Table 1 shows the numbers of maximal cliques of various sizes, the quality of each group of cliques, and the accumulated quality for each given size k. The accumulated quality for size k is produced from all maximal cliques of size k and greater. As it happens for the number of nodes in the network, the number of cliques also follows a power-law distribution. There is large overlap among cliques; for example, the two biggest maximal cliques have 16 proteins, yet they share 15 proteins. The third column in table 1 shows that the quality of maximal cliques is very high. Roughly, the bigger the maximal clique, the higher the quality of cliques. This strongly motivates us to consider maximal cliques at each level of our multilevel algorithms.

Table 1. The numbers of cliques in size, the quality of each group of cliques, and the accumulated quality. The quality was computed using the number of correctly grouped nodes and the total number of nodes belonging to maximal cliques.

Size	Number of maximal cliques	Quality of cliques	Accumulated quality
3	726	$1378/2178 = 63.27\%$	$4456/5895 = 75.59\%$
4	260	$671/1040 = 64.52\%$	$3078/3717 = 82.81\%$
5	133	$548/665 = 82.41\%$	$2407/2677 = 89.91\%$
6	118	$612/708 = 86.44\%$	$1859/2012 = 92.40\%$
7	70	$447/490 = 91.22\%$	$1247/1304 = 95.63\%$
8	38	$304/304 = 100.00\%$	$800/814 = 98.28\%$
9	32	$282/288 = 97.92\%$	$496/510 = 97.25\%$
10	8	$78/80 = 97.50\%$	$214/222 = 96.39\%$
11	2	$18/22 = 81.82\%$	$136/142 = 95.77\%$
12	5	$60/60 = 100.00\%$	$118/120 = 98.33\%$
13	0	N/A	$58/60 = 96.66\%$
14	2	$28/28 = 100.00\%$	$58/60 = 96.66\%$
15	0	N/A	$30/32 = 93.75\%$
16	2	$30/32 = 93.75\%$	$30/32 = 93.75\%$

When dealing with the TC based multilevel clustering algorithm [2], a question left unanswered is how to deal with overlaps between groups of nodes (in our approach the groups are maximal cliques). In [2] we presented four methods for TCs. When considering two TCs there are three cases of overlap, sharing none, one, or two nodes. The experimental results showed that the best accuracy are expected with TC_ONE. That is, two TCs sharing two nodes are merged and two TCs sharing one node are separated with the sharing node assigned to one of them. Zhang *et al*. [6] uses a good criterion on how to merge maximal cliques. We apply a similar criterion to our TCs and maximal cliques. We define the density of subgraphs with n nodes with e edges as $Q = 2e/(n(n-1))$. That is, the Q value of a subgraph is the fraction of the actual number of edges to the possible total number of edges. For cliques the density Q is equal to one. And for two TCs sharing two nodes, the subgraph including two TCs has five actual edges with four nodes, so $Q = 5/6 = 0.83$. Similarly, two TCs with one sharing node have $Q = 3/5 = 0.6$.

Our approach is to use this Q value as in [5, 6]. The importance of Q value has been supported in various papers [25–27]. We can set a certain cutoff value on Q value, Q_{cutoff}, to decide which cliques are merged. We will talk about more details on choosing Q_{cutoff} in section 3.1. So our approach is described as follows. The Bron–Kerbosch algorithm is used to enumerate all maximal cliques in the network [28]. The maximal cliques are sorted and stored in a *clique list* of decreasing order according to its clique size. Then, the biggest clique is compared to each with all the other cliques and the Q value of the subgraph including these two cliques is computed, when they have overlapping nodes. Subgraphs with a Q value higher than a certain Q_{cutoff} are merged. Those with lower Q values than Q_{cutoff} are not combined, instead they are modified to avoid overlap among them. They are still cliques, so these are added to the *clique list*. We repeat this process for all cliques in the *clique list* until all cliques of size two or greater are used. After we are done, all singletons in the list are left unmerged. We refer to the just described coarsening algorithm as the Maximal Clique Merging (MCM) algorithm.

So we now have to deal with the unmerged nodes in the MCM algorithm. For our graph, there are 574 unmerged nodes. In addition, using $Q_{\text{cutoff}} = 0.82$, 184 singleton nodes were moved to the group of unmerged nodes during the elimination of overlapping parts of cliques mentioned in the previous paragraph.

Since the proteins in the unmerged group do not comprise any maximal clique (for clique sizes of three or greater), it is natural to apply a matching based algorithm to the second group. There are several matching based algorithms for unweighted networks. Among them, HESN, which stands for Heavy-Edge-Small-Node, was developed in [1]. Edge weights in the original PPI networks are initially all one. But after one level of coarsening some edges in the coarsened graph may represent more than one edge (in fact, up to 4) of the original graph. 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 edge weights: merge nodes with highest edge weights. But there are many nodes with the same edge weight. Thus we give higher priority to the edge with fewer combining node weights, which is defined as the number of nodes included in the supernode. To break ties, we choose the edge maximizing $w(n_i)^{-1} + w(n_j)^{-1}$ where $w(n_i)$ and $w(n_j)$ are the node weights (the number of nodes) of supernodes n_i and n_j. This method can be applied to the nodes unmerged by the MCM algorithm.

After one level of coarsening with MCM is done, the newly created coarser network may also have maximal cliques. But it is likely that there are fewer maximal cliques after MCM is applied.

3. Results

3.1 *Optimum cutoff value*

Here we show how changes in Q_{cutoff} for MCM coarsening algorithm affects the number of nodes merged and the quality of coarsening, and we discuss how to decide the optimum Q_{cutoff}.

Let us first start with four TC based algorithms. As seen in section 2.4, the subgraph with TCs sharing one node has $Q = 0.83$ and two TCs with one sharing node produces $Q = 0.6$. TC_ALL merges all TCs sharing any node, and merges too many nodes, which is not wanted. TC_TWO picks one TC and merges any TC which shares one or two nodes, so Q_{cutoff} can be any number less than 0.6. TC_ONE similarly picks one TC and merges with TCs sharing only two nodes, so Q_{cutoff} should be between 0.6 and 0.83. Finally, TC_ONLY does not merge with any other TC, so any number greater than 0.83 is acceptable for Q_{cutoff}. We showed TC_ONE and TC_ONLY outperform TC_TWO and TC_ALL in [2]. So for TC based algorithms, Q_{cutoff} can be chosen to be between 0.6 and 1 to expect a good coarsening. The qualities of four TC based algorithms are listed for comparison in table 2. There are 6968 TCs found compared to 1396 maximal cliques. The biggest supernode in TC_ALL has 959 nodes and next biggest one has 12. As expected, the quality of this algorithm is much worse.

Table 2. The qualities and the number of supernodes with four different TC base coarsening algorithms on 2-core network which has 1717 nodes.

TC algorithm	Number of supernodes	Accuracy with size
TC_ALL	45	190 out of 1143 = 16.6%
TC_TWO	102	609 out of 917 = 66.4%
TC_ONE	142	646 out of 829 = 77.9%
TC_ONLY	233	581 out of 699 = 83.1%

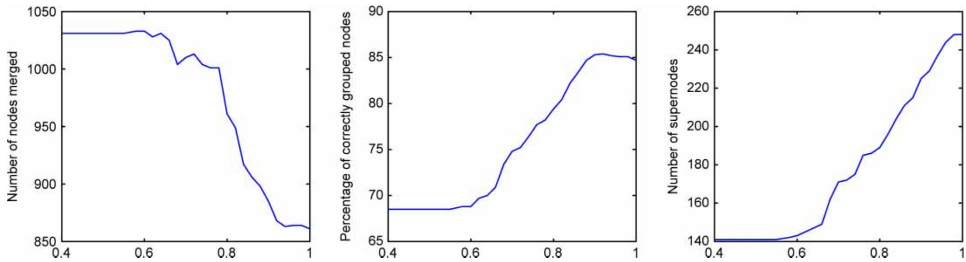

Figure 2. The number of nodes merged, the percentage of correctly grouped nodes, and the number of supernodes for $0.4 \leq Q_{\text{cutoff}} \leq 1$.

We now talk about how the quality of coarsening and the size of the coarsened network with MCM change for various Q_{cutoff}. Figure 2 shows the number of nodes merged, the percentage of correctly grouped nodes, and the number of supernodes for $0.4 \leq Q_{\text{cutoff}} \leq 1$. All three figures start with a flat line because the choice of $Q_{\text{cutoff}} \leq 0.55$ does not make a difference. As Q_{cutoff} increases, the quality of coarsening and the number of supernodes consistently increases up to some point. However, the number of distinct nodes in the supernodes keeps decreasing. This can be well explained as follows: the higher Q_{cutoff}, the fewer nodes are merged; the higher Q_{cutoff}, the higher quality of groupings; the higher Q_{cutoff}, the smaller and more numerous the supernodes. Even though coarsening algorithms aim both to merge as many nodes as possible and to maintain the quality of coarsening, no Q_{cutoff} satisfies both conditions at the same time. So we have to trade off between the number of nodes merged and the quality of coarsening. We present two viewpoints to help decide an optimal Q_{cutoff}. One is the quality point of view. Since a coarsening algorithm is not a clustering algorithm, it is better not to merge too much, as TC_ALL does. Clustering is also considered as grouping supernodes (in some sense it is merging). That is, all nodes go through one more merging stage. So, we rather go with 'less' merging than 'more' merging for higher quality of clustering. The other point of view is in terms of time. Less merging with a high Q_{cutoff} generates a big coarsened network which requires more computational time to cluster. Thus, the optimum Q_{cutoff} should be determined according to the relative importance of time and accuracy.

3.2 *Comparison of various multilevel algorithms*

Finally, we talk about the effect of the MCM algorithm and its variants when they are actually applied to the 2-core network. We compare the clustering results of plain Minmaxcut with those of a TC based multilevel algorithm, TC_ONE, and MCM and its variants. We first fix our cutoff value, Q_{cutoff}, at 0.78 for MCM for the rest of the experiments. When one level of coarsening is completed by MCM, we consider two options. The first is one more coarsening with MCM (that is, two levels of coarsening with MCM); we call this method MCM2. The other option is to use a matching based coarsening algorithm, HESN, for the second level; we call this MCM + HESN. The model network, 2-core network, does not have any more maximal cliques size of three or greater after two levels of coarsening with MCM. So we only consider MCM2 and MCM + HESN for the test. TC_ONE is also compared, along with TC_ONE2 which has two levels of TC_ONE.

One issue dealing with clustering algorithms is the number of clusters created. After clustering the 2-core network, all nodes not in 2-core(Y) are added. The 139 nodes not in 1-path(2-core(Y)) are stored in 59 additional components. These separate components can be directly added as distinct clusters. The remaining 852 nodes in 1-path(2-core(Y)) are included in existing clusters as described in section 2.2. Under the assumption that we do not have any

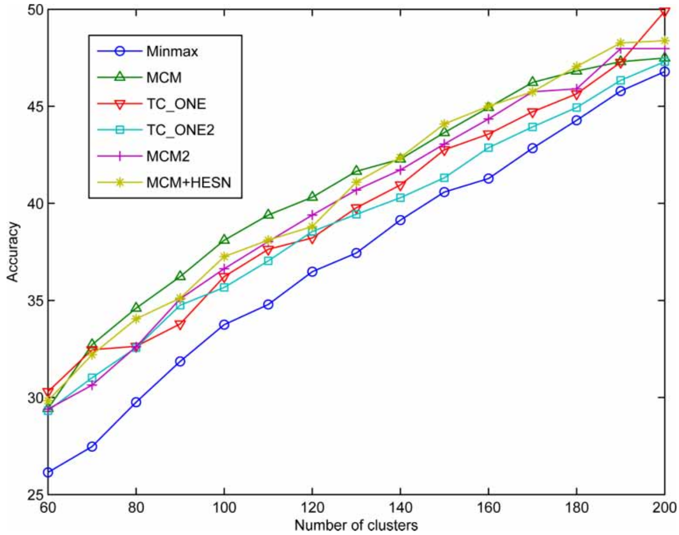

Figure 3. The accuracy of Minmaxcut and the five multilevel algorithms, MCM, TC_ONE, TC_ONE with two levels, MCM with two levels, and MCM + HESN.

information on how many clusters are supposed to be constructed, we create 60 through 200 clusters from 2-core(Y). The 59 additional components from $Y\setminus$1-path(2-core(Y)) give 119 through 259 clusters in total.

Figure 3 shows the accuracy for Minmaxcut and the five multilevel algorithms, MCM, TC_ONE, TC_ONE2, MCM2, and MCM+HESN to build 60 through 200 clusters. The accuracy of each method mostly keeps increasing as the number of clusters increases. All methods show improvements from the plain Minmaxcut method. In particular, MCM, MCM2, and MCM + HESN reliably improve the accuracy of clusters by, on average, 10% (and up to 15%). The MCM related methods mostly work better than TC_ONE. MCM usually outperforms the other multilevel algorithms in accuracy. The reason TC_ONE2 does not outperform TC_ONE is easily explained: TC_ONE2 over-merges nodes (Q_{cutoff} too low) so the coarsening becomes bad like TC_ALL.

The total elapsed time for these six methods appear on the left hand side of figure 4. Time is measured in seconds. Top right figure shows the accumulated time after each of three stages of multilevel algorithms finished for MCM method. The bottom right figure shows those for MCM + HESN. The total elapsed time for Minmaxcut is dedicated to partitioning. Also, the number of clusters does not increase the total time much. Of course, there should be much difference from a few levels to 10. This means that after some number of clusters are obtained by Minmaxcut, constructing extra clusters does not take much time. The middle lines in (b) and (c) of figure 4 indicate the time consumed for coarsening and partitioning. Those are all very flat like plain Minmaxcut. This leads to the conclusion that the total time for these multilevel algorithms depends mostly on the refinement step. This phenomenon occurs for all other multilevel algorithms. Since KL refinement algorithm (complexity $O(N^2)$) performs poorly with a large number of clusters, it was modified to generate good results. Coarsening time for MCM is 1.16 seconds compared to 55.69 seconds for partitioning. Only 0.55 seconds is used to enumerate all maximal cliques. Even though finding all maximal cliques in a network is NP-hard, it doesn't take much time to detect them in sparse networks like PPI networks.

Of the five multilevel methods considered, MCM + HESN takes the least time in general.

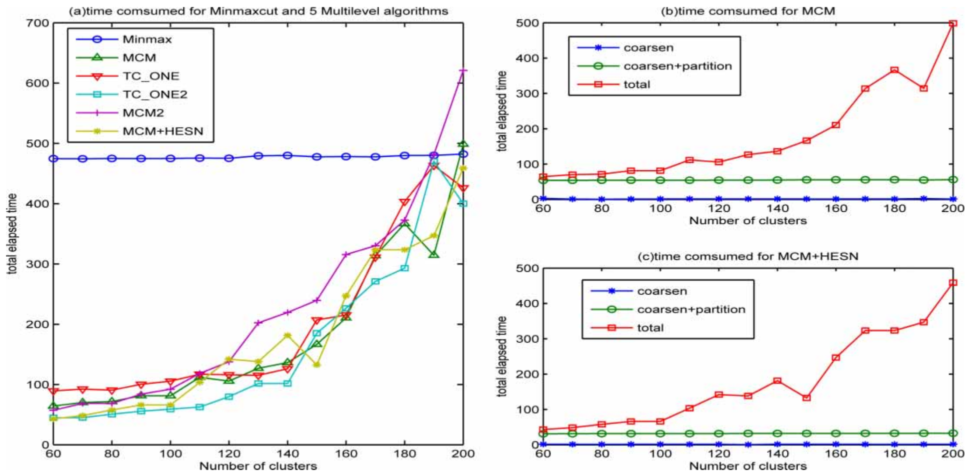

Figure 4. (a) The total elapsed time of Minmaxcut and the five multilevel algorithms, MCM, TC_ONE, TC_ONE with two levels, MCM with two levels, and MCM + HESN; (b) time for MCM with three multilevel steps, coarsening, partitioning, and refining; (c) time for MCM + HESN with three steps. Time is measured in seconds.

4. Conclusion

Traditional matching based multilevel algorithms which merge at most two nodes at a time to construct a supernode are limited, especially on unweighted graphs, because they do not use any structured analysis. We present triangular clique (TC) based multilevel algorithms and further maximal clique merging multilevel algorithms to overcome this problem. Qualities of supernodes with our clique merging multilevel algorithms are better even though time complexities are much bigger than other matching based algorithms because all maximal cliques should be enumerated in advance. Fortunately, all maximal cliques are enumerated pretty rapidly on scale free networks. Most of the time is spent at the refining step, especially when dealing with many clusters.

Acknowledgements

We thank all the referees for some corrections and many suggestions about the presentation of this research. We also thank Joe Eichholz and David Stewart for proofreading the last draft. This work was supported in part by NSF ITR grant DMS-0213305.

References

[1] Oliveira, S. and Seok, S.C., 2006, A multilevel approach for identifying functional modules in protein–protein interaction networks. *Proceedings of IWBRA 2006*, Lecture Notes in Computer Science, 3992 (Berlin: Springer-Verlag), pp. 726–773.

[2] Oliveira, S. and Seok, S.C., 2006, Triangular clique based multilevel approaches to identify functional modules. Paper presented at the Seventh International Meeting on High Performance Computing for Computational Science (VECPAR 2006) 10–13 July 2006, Rio de Janeiro, Brazil.

[3] Oliveira, S. and Seok, S.C., A matrix-based multilevel approach to identify functional protein models. *International Journal of Bioinformatics Research and Applications*. To appear.

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

[5] Spirin, V. and Mirny, L.A., 2003, Protein complexes and functional modules in molecular networks. *Proc. Natl. Acad. Sci. USA*, **100**, 12123–12128.

[6] Zhang, C., Liu, S. and Zhou, Y.Q., 2006, Fast and accurate method for identifying high-quality protein-interaction modules by clique merging and its application to yeast. *J. Proteome Res.*, **5**, 801–807.

[7] Krogan, N.J., Cagney, G., Yu, H.Y., Zhong, G.Q., Guo, X.H., Ignatchenko, A., Li, J., Pu, S.Y., Datta, N., Tikuisis, A.P., Punna, T., Peregrin-Alvarez, J.M., Shales, M., Zhang, X., Davey, M., Robinson, M.D., Paccanaro, A., Bray, J.E., Sheung, A., Beattie, B., Richards, D.P., Canadien, V., Lalev, A., Mena, F., Wong, P., Starostine, A., Canete, M.M., Vlasblom, J., Wu, S., Orsi, C., Collins, S.R., Chandran, S., Haw, R., Rilstone, J.J., Gandi, K., Thompson, N.J., Musso, G., St Onge, P., Ghanny, S., Lam, MH.Y., Butland, G., Altaf-Ui, A.M., Kanaya, S., Shilatifard, A., O'Shea, E., Weissman, J.S., Ingles, C.J., Hughes, T.R., Parkinson, J., Gerstein, M., Wodak, S.J., Emili, A. and Greenblatt, J.F., 2006, Global landscape of protein complexes in the yeast saccharomyces cerevisiae. *Nature*, **440**, 637–643.

[8] Bornholdt, S. and Schuster, H.G. (Eds), 2003, *Handbook of Graphs and Networks* (Wilhelm: Wiley VCH).

[9] Uetz, P., Giot, L., Cagney, G., Mansfield, T.A., Judson, R.S., Knight, J.R., Lockshon, D., Narayan, V., Srinivasan, M., Pochart, P., Qureshi-Emili, A., Li, Y., Godwin, B., Conover, D., Kalbfleisch, T., Vijayadamodar, G., Yang, M.J., Johnston, M., Fields, S. and Rothberg, J.M., 2000, A comprehensive analysis of protein–protein interactions in saccharomyces cerevisiae. *Nature*, **403**, 623–627.

[10] Ito, T., Chiba, T., Ozawa, R., Yoshida, M., Hattori, M. and Sakaki, Y., 2001, A comprehensive two-hybrid analysis to explore the yeast protein interactome. *Proc. Natl. Acad. Sci.*, **98**, 4277–4278.

[11] Gavin, A.C., Bosche, M., Krause, R., Grandi, P., Marzioch, M., Bauer, A., Schultz, J., Rick, J.M., Michon, A.M., Cruciat, C.M., Remor, M., Hofert, C., Schelder, M., Brajenovic, M., Ruffner, H., Merino, A., Klein, K., Hudak, M., Dickson, D., Rudi, T., Gnau, V., Bauch, A., Bastuck, S., Huhse, B., Leutwein, C., Heurtier, M.A., Copley, R.R., Edelmann, A., Querfurth, E., Rybin, V., Drewes, G., Raida, M., Bouwmeester, T., Bork, P., Seraphin, B., Kuster, B., Neubauer, G. and Superti-Furga, G., 2002, Functional organization of the yeast proteome by systematic analysis of protein complexes. *Nature*, **415**, 141–147.

[12] Ho, Y., Gruhler, A., Heilbut, A., Bader, G.D., Moore, L., Adams, S.L., Millar, A., Taylor, P., Bennett, K., Boutilier, K., Yang, L.Y., Wolting, C., Donaldson, I., Schandorff, S., Shewnarane, J., Vo, M., Taggart, J., Goudreault, M., Muskat, B., Alfarano, C., Dewar, D., Lin, Z., Michalickova, K., Willems, A.R., Sassi, H., Nielsen, P.A., Rasmussen, K.J., Andersen, J.R., Johansen, L.E., Hansen, L.H., Jespersen, H., Podtelejnikov, A., Nielsen, E., Crawford, J., Poulsen, V., Sorensen, B.D., Matthiesen, J., Hendrickson, R.C., Gleeson, F., Pawson, T., Moran, M.F., Durocher, D., Mann, M., Hogue, C.W.V., Figeys, D. and Tyers, M., 2002, Systematic identification of protein complexes in saccharomyces cerevisiae by mass spectrometry. *Nature*, **415**, 123–124.

[13] Mewes, H.W., Amid, C., Arnold, R., Frishman, D., Guldener, U., Mannhaupt, G., Munsterkotter, M., Pagel, P., Strack, N., Stumpflen, V., Warfsmann, J. and Ruepp, A., 2004, Mips: Analysis and annotation of proteins from whole genomes. *Nucleic Acids Res.*, **32**, D41–4.

[14] Von Mering, C., Krause, R., Snel, B., Cornell, M., Oliver, S.G., Fields, S. and Bork, P., 2002, Comparative assessment of large-scale data sets of protein–protein interactions. *Nature*, **417**, 399–403.

[15] Seok, S.C., 2007, *Multilevel Clustering Algorithms for Documents and Large-Scale Proteomic Networks*. PhD thesis. In preparation.

[16] Guldener, U., Munsterkotter, M., Kastenmuller, G., Strack, N. van Helden, J., Lemer, C., Richelles, J., Wodak, S.J., Garcia-Martinez, J., Perez-Ortin, J.E., Michael, H., Kaps, A., Talla, E., Dujon, B., Andre, B., Souciet, J.L., De Montigny, J., Bon, E., Gaillardin, C. and Mewes, H.W., 2005, Cygd: the comprehensive yeast genome database. *Nucleic Acids Research*, **33**, D364–D368. Database issue.

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

[18] Bader, G.D. and Hogue, C.W., 2003, An automated method for finding molecular complexes in large protein interaction networks. *BMC Bioinformatics*, **4**, article 2.

[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, 3695, pp. 265–276.

[20] Ding, C., He, X., Zha, H., Gu, M. and Simon, H., 2001, A min-max cut algorithm for graph partitioning data clustering. In *ICDM 2001, Proceedings IEEE International Conference on Data Mining, 2001*, pages 107–114.

[21] Ding, C., He, X., Meraz, R.F. and Holbrook, S.R., 2004, A unified representation of multiprotein complex data for modeling interaction networks. *Proteins: Structure, Function, and Bioinformatics*, **57**, 99–108.

[22] Kernighan, B.W. and Lin, S., 1970, An efficient heuristic procedure for partitioning graphs. *The Bell System Technical Journal*, **49**, 291–307.

[23] Fiduccia, C.M. and Mattheyses, R.M., 1982, A linear time heuristic for improving network partitions. 19th IEEE Design Automation Conference, pages 175–181.

[24] Skiena, S., 1998, *The Algorithm Design Manual* (New York: Springer-Verlag).

[25] Watts, D.J. and Strogatz, S.H., 1998, Collective dynamics of 'small-world' networks. *Nature*, **393**, 409–410.

[26] Goldberg, D.S. and Roth, F.P., 2003, Assessing experimentally derived interactions in a small world. *Proceedings of the National Academy of Sciences*, **100**, 4372–4376.

[27] Barabasi, A.L. and Oltvai, Z.N., 2004, Network biology: Understanding the cell's functional organization. *Nature Reviews Genetics*, **5**, 101–113.

[28] Bron, C. and Kerbosch, J., 1973, Algorithm 457: Finding all cliques of an undirected graph. *Communications of the ACM*, **16**, 575–577.