

# A Parallel Algorithm for Clustering Protein-Protein Interaction Networks

Qiaofeng Yang  
Department of Computer Science  
University of California  
Riverside, CA 92521, USA  
qyang@cs.ucr.edu

Stefano Lonardi  
Department of Computer Science  
University of California  
Riverside, CA 92521, USA  
stelo@cs.ucr.edu

## Abstract

*The increasing availability of interaction graphs requires new resource-efficient tools capable of extracting valuable biological knowledge from these networks. In this paper we report on a novel parallel implementation of Girvan and Newman's clustering algorithm that is capable of running on clusters of computers. Our parallel implementation achieves almost linear speed-up up to 32 processors and allows us to run this computationally intensive algorithm on large protein-protein interaction networks. Preliminary experiments show that the algorithm has very high accuracy in identifying functional related protein modules.*

*Software will be made available in the public domain at <http://www.cs.ucr.edu/~qyang/>*

## 1 Introduction

Recent advances in proteomics such as yeast two-hybrid, phage display and mass spectrometry have resulted in several genome-scale protein-protein interaction (PPI) map projects. The identification of functional related proteins is among the most urgent computational challenges facing the scientific community. In the literature, the problem has been approached by analyzing the topological properties of interaction networks (see, e.g., [2, 11]) or by comparing networks from several model organisms (see, for example, [8, 9, 12]).

In [2], Bader and Hogue described a graph theoretic clustering algorithm for finding potential protein complexes in large PPI networks. The method is based on vertex weighting by local neighborhood density and outward traversal from a locally dense seed protein to isolate the dense regions according to given parameters. Rives and Galitski [11] studied modular organization

of cellular networks by developing a network clustering method. An all-pairs shortest path distance matrix is constructed and transformed into an association matrix with each entry defined as  $1/d^2$ , where  $d$  is the shortest path distance between two vertices in the network. Hierarchical agglomerative average-linkage clustering with the uncentered correlation coefficient as the distance metric is then applied to the association matrix.

With the availability of PPI networks from several model organisms, recent research on identifying functional related proteins has focused comparative approaches. Kelley et. al. [8] have employed ideas from sequence alignment and applied them on PPI networks of budding yeast *S.cerevisiae* and bacterial pathogen *H. pylori*. Conserved pathways are identified by global aligning two PPI networks. To perform the alignment, the two networks are combined into a global alignment graph in which each vertex represents a pair of proteins from each network which are similar at sequence level and each edge represents a conserved interaction, gap, or mismatch. A log probability score is formulated over vertices and edges of a path in the global alignment graph. The highest-scoring path is found by dynamic programming after decomposing the global alignment graph to acyclic graphs. A similar approach was adopted by Sharan et. al. [12] to identify protein complexes by comparing two PPI networks. An *orthology graph*, similar to global alignment graph, is constructed where edges are assigned weights so that high weighted sub-graphs correspond to conserved protein complexes. A heuristic algorithm is proposed based on forming high weighted seeds, refining them by exhaustive enumeration, and then expanding them using local search. Koyuturk et. al. [9] introduced the idea of pairwise local alignment of PPI networks based on an evolution model of PPI network. The proteins in two networks are classified as orthologs and paralogs based on sequence similarity. An alignment graph is constructed on orthologous

and paralogous proteins with edges classified as *match*, *mismatch*, and *duplication*. Each type of edge is associated with a reward or penalty so that the maximum weight-induced subgraph corresponds the best alignment between two networks. The maximum weight-induced subgraph is found by seeding a subgraph with a protein that has a large number of conserved interactions and small number of mismatched interactions and growing it by adding proteins that share conserved interactions with this subgraph one by one.

Here we are interested in discovering functional related proteins by clustering interaction graphs based on their topological properties. Proteins that are involved in the same cellular process or reside in the same protein complex are expected to have strong interactions with their partners. At the same time, interactions between distinct functional modules are expected to be suppressed in order to increase the overall robustness of the network by localizing effects of deleterious perturbations [10]. Therefore, the identification of densely connected subgraphs in PPI networks may reveal functional related protein modules.

Among the wide spectrum of graph clustering algorithms available in literature, we selected the clustering algorithm by Girvan and Newman [5], which showed impressive performances in discovering community structures in several networks, such as social networks, scientific collaborations, food web, etc. Qualitatively, a community is defined as a subset of vertices within the graph such that connections between the vertices are denser than connections with the rest of the network. To the best of our knowledge Girvan and Newman's algorithm has never been used on interaction networks. One of the reasons may be due to its high computational cost. In order to run it on large interaction graphs we devised a parallel implementation which achieves almost linear speed-up up to 32 processors. Preliminary experiments on several protein-protein interaction networks show that it is very effective in identifying functional related protein modules.

## 2 Edge Betweenness Clustering

Girvan and Newman's [5] algorithm is a novel divisive clustering algorithm for graphs. In divisive algorithms, one starts with the whole graph and iteratively removes the edges, thus dividing the network progressively into smaller and smaller disconnected subnetworks. In Girvan and Newman's algorithm the edges are removed based on the value of their *edge betweenness*, which is a generalization of the *centrality betweenness* first introduced by Anthonisse [1] and Freeman [4].

Consider the shortest paths between all pairs of ver-

tices in a graph. The *betweenness* of an edge is defined as the number of these paths running through it. When a graph is made of tightly intra-connected and loosely inter-connected clusters, all shortest paths between vertices in distinct clusters have to traverse the few inter-cluster connections, which therefore have a high betweenness value. By removing those edges first, the clusters are separated from one another, thus revealing the underlying community structure of the graph. Girvan and Newman's clustering algorithm works as follows: (1) Calculate the betweenness for all edges in the network; (2) Remove the edge with the highest betweenness; (3) Recalculate the betweenness for all edges affected by the removal; (4) Repeat from step 2 until no edge remains.

A single step of the edge betweenness algorithm consists of the computation of the edge betweenness values for all the edges in the graph and the removal of the edge with the highest value. The iterative removal of edges leads to the decomposition of the network into disconnected subgraphs which in their turn undergo the same procedure, until the whole graph is divided into a set of isolated vertices. The relationship of the disconnected subgraphs in the decomposition process can be represented by a hierarchical tree which is built based on the reverse order of the removal of the edges.

Girvan and Newman's clustering algorithm is computationally expensive. Evaluating the betweenness value for all edges requires  $O(nm)$  time, where  $n$  is the number of vertices and  $m$  the number of edges in the graph. The iterative removal of all  $m$  edges leads a worst-case time complexity of  $O(nm^2)$ , which makes the algorithm practically unfeasible for large networks.

## 3 Modularity

The output of the edge betweenness clustering algorithm described above is a hierarchical tree representing the clusters. Clearly the algorithm will produce such a tree even if the input graph is random, although in this case the clusters will not be very meaningful. We need, therefore, an objective measure of the quality of the clusters. Newman and Girvan measure the quality of the clusters found by the algorithm by introducing the concept of *modularity* [6]. In the following, we use the same notation as in [6]. Decompose the network into  $k$  communities. Construct a symmetric matrix  $\mathbf{e}$  of size  $k \times k$ . An element  $e_{ij}$  in  $\mathbf{e}$  represents the fraction of all edges in the network that link vertices in community  $i$  to vertices in community  $j$ . The trace of this matrix  $\text{Tre} = \sum_i e_{ii}$  represents the fraction of edges in the network that connect vertices in the same community. Summation of row (or column) elements  $a_i = \sum_j e_{ij}$  represents the frac-

tion of edges that connect to vertices in community  $i$ . The *modularity* is defined as

$$Q = \sum_i (e_{ii} - a_i^2) = \text{Tr} \mathbf{e} - \|\mathbf{e}^2\|$$

As pointed out in [6], this quantity measures the fraction of the edges in the network that connect vertices of the same type (i.e., intra-community edges) minus the expected value of the same quantity in a network with the same community divisions but random connections between the vertices. According to [6], for a random network,  $Q$  approaches 0. Values approaching  $Q = 1$ , which is the maximum, indicate strong community structure. The higher the value is, the stronger the community structure is. In section of experimental results, we will show the clustering results using modularity measure defined above on five PPI networks.

## 4 Parallel Edge Betweenness Clustering

In our parallel algorithm we exploited the ideas in [3], where the authors propose a new accumulation technique that integrates well with traversal algorithms, particularly breadth-first search, for solving the single-source shortest path problem from the source to all other vertices. By finding all-pairs shortest paths using breadth-first search starting from each vertex in the graph, the edge betweenness value can be obtained by summing pair-dependencies over all the traversals for each edge. The pair-dependency is defined as  $\delta_{st}(v) = \sigma_{st}(v)/\sigma_{st}$ , where  $\sigma_{st}$  denotes the number of shortest paths from  $s \in V$  to  $t \in V$  and  $\sigma_{st}(v)$  is the number of shortest paths from  $s$  to  $t$  which go through  $v$ . Pair-dependencies calculated from each BFS for every vertex in the graph are additive, from which edge betweenness value can be obtained. Since BFS can be performed independently and simultaneously from each vertex in the graph, the calculation required at each iteration of finding the edge with the highest betweenness value can be done by parallelizing all-pairs shortest paths. The parallel algorithm is sketched in Figure 1. Vertices in the graph are evenly assigned to all the processors. Each processor has its own copy of the graph. The procedure is initiated by a host processor. Each processor performs BFS from all the vertices assigned to it and sums up local pair-dependencies obtained from each BFS. The local pair-dependencies are then sent to the host processor. The host processor is responsible for summing up all the local pair-dependencies from each processor, obtaining the global pair-dependencies, and finding the edge with the highest betweenness value. The edge with the highest betweenness value is then broadcast by the host processor to all the processors in the communication world.

**Input:** Graph  $G$

**Output:** A list of edges in the reverse removal order

1. Evenly assign the vertices in  $G$  to all processors
2. **while** the number of edges in  $G > 0$  on all processors **do**
3.     **for\_all** the vertices  $v \in G$  **do in parallel**
4.         Breadth-First-Search( $G, v$ )
5.         Send all pair-dependencies  $\delta_{st}(v)$ ,  $v \in G$  to host
6.     **end for\_all**
7.     Receive  $\delta_{st}(v)$
8.     Calculate betweenness values for all edges in  $G$
9.     Broadcast the edge  $e$  with the highest betweenness
10.     Remove edge  $e$  from  $G$  on all processors
11.     SYNCHRONIZE()
12. **end while**

**Figure 1. Sketch of the parallel edge betweenness clustering algorithm**

All the processors delete the edge received in their own graph copy and start the next iteration until no edges left in the graph.

## 5 Implementation and Experimental Results

We implemented the parallel algorithm using C in conjunction with the LAM 7.1.1 implementation of Message Passing Interface (MPI). The algorithm was tested on the Linux cluster at the Bioinformatics Core Facility at UC, Riverside. The cluster consists of thirty two dual processor Athlon MP 2800 nodes with 1GB of RAM each. Host bus clock speed is 266MHz.

Five different PPI networks downloaded from DIP database [13] were used. We ran the algorithm on the largest component in the network. The size of the largest component in each of the datasets is summarized in Table 1. The parallel edge betweenness clustering algorithm was run on each of the five datasets using the modularity value [6] as an indicator for the quality of the clusters. Table 1 summarizes the number of clusters in each network when the modularity value reaches its maximum. We used the web-based tool PANDORA [7] to annotate the clusters obtained from the algorithm. The annotation for the clusters with the highest modularity value in yeast PPI network is shown in Table 2. The results show strong functional correlations among the proteins in the same cluster using SwissProt annotation database. For example, the first cluster has 290 proteins of which 269 are annotated in SwissProt database. Most of the proteins in the first cluster are involved in ribosome and protein biosynthesis.

Figure 2 shows the speed-up of the parallel algorithm

Organism	$n$	$m$	$C$	$Q$
<i>D. melanogaster</i>	6926	20745	962	0.31
<i>S. cerevisiae</i>	4687	15138	371	0.44
<i>C. elegans</i>	2386	3825	94	0.60
<i>H. pylori</i>	686	1351	56	0.45
<i>H. sapiens</i>	563	870	17	0.80

**Table 1. Dataset summary.**  $n$  and  $m$  are the number of vertices and edges.  $C$  is the number of clusters.  $Q$  is the modularity.

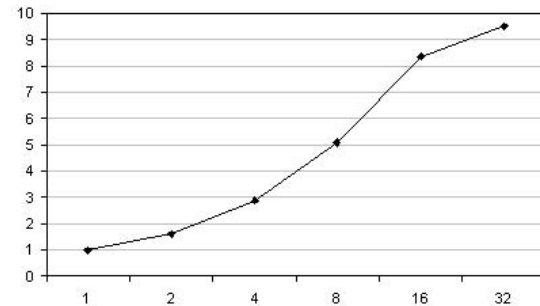
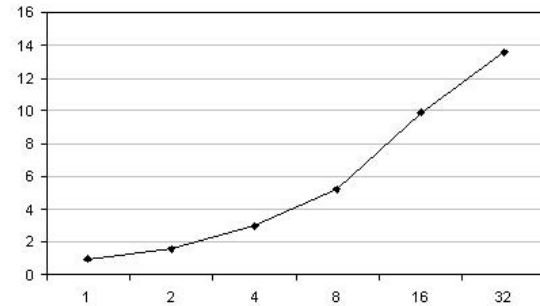
C	Size	Function assignment
1	290	Ribosome and protein biosynthesis
2	258	Protein catalytic activity related metabolism
3	130	mRNA processing
4	116	Transcription regulation
5	90	Proteasome complex and protein metabolism
6	86	Nuclear transport
7	62	Small GTPase mediated signal transduction
8	50	RNA polymerase and transcription regulation
9	43	Golgi stack and protein transport
10	36	Ribosomal proteins

**Table 2. Annotations of the clusters with the highest modularity values in *S. cerevisiae* PPI network**

over the sequential algorithm on 1, 2, 4, 8, 16, 32 processors. The speed-up is close to linear for up to 32 processors. The parallel implementation makes it possible to run the clustering algorithm on a graph of 7,000 vertices and 20,000 edges in less than 7 hours if run on 16 processors, in less than 5 hours if run on 32 processors, which would take almost 3 days if run on a single processor.

## References

- [1] J. M. Anthonisse. The rush in a directed graph. Technical report, Stichting Mathematisch Centrum, Amsterdam, 1971.
- [2] G. D. Bader and C. W. V. Hogue. An automated method for finding molecular complexes in large protein interaction networks. *BMC Bioinformatics*, 4, 2003.
- [3] U. Brandes. A faster algorithm for betweenness centrality. *Journal of Mathematical Sociology*, 25:163–177, 2001.
- [4] L. C. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, 40:35–41, 1977.
- [5] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *PNAS*, 99(12):7821–7826, 2002.
- [6] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *arXiv:cond-mat/0308217*, 1, 2003.
- [7] N. Kaplan, A. Vaaknin, and M. Linial. Pandora: Keyword-based analysis of protein sets by integration of annotation sources. *Nucleic Acids Research*, 31:5617–5626, 2003.
- [8] B. P. Kelley, R. Sharan, R. M. Karp, T. Sittler, D. E. Root, and B. R. Stockwell. Conserved pathways within bacteria and yeast as revealed by global protein network alignment. *PNAS*, 100(20):11394–11399, 2003.
- [9] M. Koyuturk, A. Grama, and W. Szpankowski. Pairwise local alignment of protein interaction networks guided by models of evolution. *RECOMB*, 2005.
- [10] S. Maslov and K. Sneppen. Specificity and stability in topology of protein networks. *Science*, 296:910–913, 2002.
- [11] A. W. Rives and T. Galitski. Modular organization of cellular networks. *PNAS*, 100(3):1128–1133, 2003.
- [12] R. Sharan, T. Ideker, B. P. Kelley, R. Shamir, and R. M. Karp. Identification of protein complexes by comparative analysis of yeast and bacterial protein interaction data. *RECOMB*, pages 282–289, 2004.
- [13] L. Xenarios, L. Salwinski, X. J. Duan, P. Higney, S.-M. Kim, and D. Eisenberg. Dip, the database of interacting proteins: a research tool for studying cellular networks of protein interactions. *Nucleic Acids Research*, 30:303–305, 2002.



**Figure 2. Speed-up on *D. melanogaster*(top) and *S. cerevisiae*(bottom) PPI networks. The  $x$ -axis is the number of processors and the  $y$ -axis is the speed-up.**