A parallel edge-betweenness clustering tool for Protein-Protein Interaction networks

Qiaofeng Yang and Stefano Lonardi*

Department of Computer Science and Engineering,
University of California, Riverside, CA 92521, USA
Fax: 1-951-827-4643 E-mail: qyang@cs.ucr.edu
E-mail: stelo@cs.ucr.edu
*Corresponding author

Abstract: The increasing availability of protein-protein interaction graphs (PPI) requires new efficient tools capable of extracting valuable biological knowledge from these networks. Among the wide range of clustering algorithms, Girvan and Newman's edge betweenness algorithm showed remarkable performances in discovering clustering structures in several real-world networks. Unfortunately, their algorithm suffers from high computational cost and it is impractical for inputs of the size of large PPI networks. Here we report on a novel parallel implementation of Girvan and Newman's clustering algorithm that achieves almost linear speed-up for up to 32 processors. The tool is available in the public domain from the authors' website.

Keywords: system biology; Protein-Protein Interaction networks; PPI; clustering of graphs; distributed tool; data mining; bioinformatics.

Reference to this paper should be made as follows: Yang, Q. and Lonardi, S. (2007) 'A parallel edge-betweenness clustering tool for Protein-Protein Interaction networks', *Int. J. Data Mining and Bioinformatics*, Vol. 1, No. 3, pp.241–247.

Biographical notes: Qiaofeng Yang received her BS in Biotechnology and MS in Biochemistry and Molecular Biology at Shanghai Jiao Tong University, P.R. China. She is a PhD candidate at the Graduate Program in Genetics, Genomics and Bioinformatics, University of California, Riverside. Her research interests include computational molecular biology, data mining, data compression, pattern discovery and advanced data structures and algorithms.

Stefano Lonardi is Assistant Professor in the Department of Computer Science and Engineering, University of California, Riverside, CA. He received his PhD from the Department of Computer Sciences, Purdue University, West Lafayette, IN. He also holds a Doctorate Degree from the University of Padua, Italy. His research interests include computational molecular biology, data mining and data compression. He has published in several major theoretical computer science and computational biology journals and conferences. In the year 2005, he received the CAREER award from National Science Foundation.

1 Introduction

Recent advances in proteomics such as yeast two-hybrid, phage display and mass spectrometry have resulted in several genome-scale PPI map projects. The identification of functionally related proteins is among the most urgent computational challenges facing the proteomics community. In the literature, the problem has been approached by analysing the topological properties of interaction networks (see, e.g., Bader and Hogue, 2003; Rives and Galitski, 2003) or by comparing networks from several model organisms (see, e.g., Kelley et al., 2003; Koyuturk et al., 2005; Sharan et al., 2004).

Here, we are interested in discovering functionally related proteins by clustering interaction graphs based on their topological properties. It has been shown (see, e.g., Maslov and Sneppen, 2002) that 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 localising effects of deleterious perturbations. Such characteristic network organisation motivates to use an algorithm in the divisive class of clustering algorithms, which discover and break down the relatively few links between different functional modules, thus revealing the clustering structure in the network.

Among the wide spectrum of graph clustering algorithms available in the literature, we selected the algorithm by Girvan and Newman (2002), which showed remarkable performances in discovering clustering structures in several networks, such as social networks, scientific collaborations, food web, and PPI networks (Dunn et al., 2005). Girvan and Newman's algorithm is a novel divisive clustering algorithm for graphs, which iteratively removes the edges of the graph, thus dividing the network progressively into smaller and smaller disconnected subgraphs. Our implementation also incorporates their computation of the modularity (Newman and Girvan, 2004) to assess the quality of the clusters.

Unfortunately, the high computational cost of Girvan and Newman's clustering algorithm has been an obstacle to its use on relatively large graphs, such as large PPI networks. In fact, the computational cost of the algorithm is already prohibitive when the input is a graph with a few thousand edges. Here we report on a parallel implementation of the algorithm, which allows users to analyse large PPI networks on a distributed cluster of computers. Experimental results show that our implementation achieves almost linear speed-up up to 32 processors. Our tool would be useful even for users that do not have access to a cluster of computers, but happen to own a dual processor computer. Preliminary experiments on several PPI networks show that it is effective in identifying clusters corresponding to functional related protein modules.

2 Implementation

For completeness of the presentation, we first briefly review Girvan and Newman's clustering algorithm. Because of its high computational cost on large PPI networks, we devise a parallel implementation of their clustering algorithm, which is discussed in detail next. Our implementation incorporates the computation of the *modularity* (Newman and Girvan, 2004) that assesses the quality of the clusters obtained by the algorithm.

2.1 Edge betweenness clustering

Given the input graph to be clustered, consider the shortest paths between all pairs of vertices in the graph. The *betweenness* of an edge is defined as the number of these paths running through it. When the graph is made of densely 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 clustering structure in 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.

Girvan and Newman's 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.

2.2 Parallel edge betweenness clustering

First, we observe that by finding all-pairs shortest paths using Breadth-First Search (BFS) starting from each vertex in the graph, the edge betweenness value can be obtained by summing pair-dependencies (Brandes, 2001) over all the traversals. The pair-dependency is defined as $\delta_{st}(v) = \sigma_{st}(v)/\sigma_{st}$, where σ_{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. Summations from all traversals will give us the overall vertex betweenness, from which edge betweenness can be obtained by a trivial generalisation. 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 parallelising all-pairs shortest paths. The parallel algorithm is sketched in Figure 1.

The vertices of the graph are evenly assigned to all the processors, but each processor has its own copy of the graph. The procedure is initiated by a host processor, and then each processor performs BFS from all the vertices assigned to it and sums up partial pair-dependencies obtained from each BFS. The partial pairdependencies are then sent to the host processor. The host processor is responsible for summing up all the partial 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. All the processors delete the edge received in their own graph copy and start the next iteration until no edges are left in the graph.

Figure 1 Sketch of the parallel edge betweenness clustering algorithm

```
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
```

2.3 *Modularity*

The output of Girvan and Newman's betweenness algorithm is the order of removal of the edges, which implicitly defines a hierarchical tree on the nodes of the graph. In order to determine where to cut the tree to create the clusters, the notion of modularity is used.

Suppose there are k clusters in the current iteration of the algorithm. A symmetric matrix E of size $k \times k$ is constructed as follows. An element e_{ij} in E represents the fraction of all edges that link the vertices in cluster i to the vertices in cluster j and e_{ii} represents the fraction of edges that connect vertices within cluster i. Thus, summation of row (or column) elements $c_i = \sum_{j=1}^{k} e_{ij}$ represents the fraction of all edges that connect vertices to and within cluster i.

The *modularity* is then defined as $Q = \sum_{i=1}^{k} (e_{ii} - c_i^2)$, which measures the fraction of the edges that connect vertices within the same cluster minus the expected value of the same quantity in the network (Newman and Girvan, 2004). For a random network with random decomposition, Q approaches 0. Values approaching $Q = 1$, which is the maximum, indicate strong clustering structure. The higher is the value, the stronger is the clustering structure in the network.

2.4 *Platforms*

The parallel edge betweenness clustering tool was written in C++ under Linux. The minimum requirement for the software is LAM (7.1.1 preferred) and the Boost Graph Library (both of which are in the public domain). In principle, any platforms on which LAM/MPI can be installed and have a gcc compiler can compile and run our tool. The implementation was extensively tested on the Linux cluster at the Bioinformatics Core Facility at UC, Riverside. The cluster consists of 32 dual processor Athlon MP 2800 nodes with 1 GB of RAM each.

2.5 Usage

The tool does not require any parameter other than the input filename. The input file must be formatted as an edge list of the PPI network, in which each pair of interactions between two proteins is listed on a single line. For example, one line of the input file may look like `protein_name1 protein_name2` (separated by a space). The computation is carried out on the largest connected component of the network. The software outputs all the clusters at the point where the modularity value reaches maximum. Clusters are indexed by an integer ID, which is followed by a list of the proteins which belong to the same cluster.

3 Results and discussion

Five different PPI networks downloaded from DIP database (Xenarios et al., 2002) 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 summarised in Table 1. The parallel edge betweenness clustering algorithm was run on each of the five datasets using the modularity value (Newman and Girvan, 2004) as an indicator for the quality of the clusters. Table 1 summarises the number of clusters in each network when the modularity value reaches its maximum. We used the web-based tool Pandora (Kaplan et al., 2003) to annotate the clusters obtained from the algorithm. The annotation for the clusters with the overall 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 fifth cluster has 116 proteins of which 101 are annotated in SwissProt database. Most of the proteins in the fifth cluster are involved in transcription regulation.

Figure 2 shows the speed-up of the parallel implementation of the edge between ness calculation over the sequential algorithm on 1, 2, 4, 8, 16, 32 processors. The speed-up is close to linear for up to 32 processors for two largest PPI networks (*D. melanogaster* and *S. cerevisiae*). 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 three days if run on a single processor.

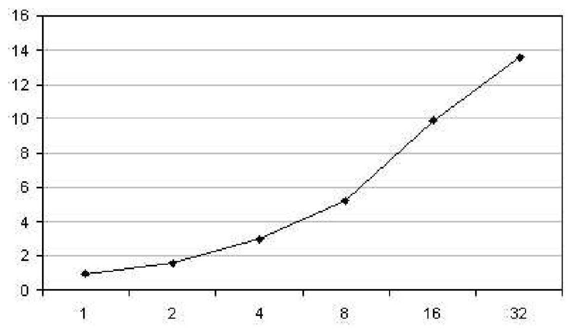
Table 1 Dataset summary *n* and *m* are the number of vertices and edges, respectively. *C* is the number of clusters produced. *Q* is the value of the modularity

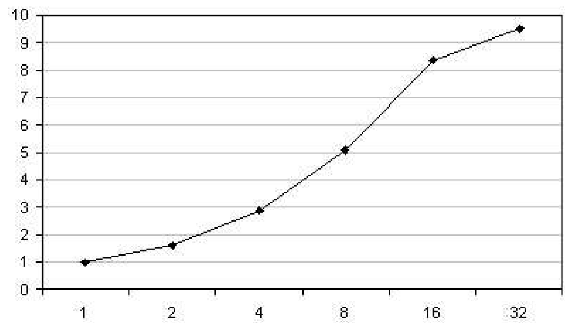
Organism	*n*	*m*	*C*	*Q*
D. melanogaster	6926	20745	914	0.36
S. cerevisiae	4687	15138	342	0.48
C. elegans	2386	3825	81	0.63
H. pylori	686	1351	45	0.50
H. sapiens	563	870	17	0.82

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

C	*Size*	*Function assignment*
1	652	Cell cycle
2	316	Protein biosynthesis
3	152	Transmembrane proteins
4	130	mRNA splicing
5	116	Transcription regulation
6	86	Nuclear transport
7	86	mRNA-processing
8	83	Hypothetical proteins
9	79	Mitochondrion transmembrane proteins
10	68	Hydrolases and transferases

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





4 Conclusions

Our tool is a practical software tool that allows the exploration of clustering structures in large PPI graphs (and potentially in other large biological network). The tool is designed for the efficient utilisation of the computational resources available in distributed cluster of computers.

Acknowledgements

This project was supported in part by NSF CAREER IIS-0447773 and NSF DBI-0321756. We thank Dr. M. Newman for providing us the sequential implementation of the edge betweenness clustering algorithm and for helpful discussions.

References

Bader, G. and Hogue, C. (2003) 'An automated method for finding molecular complexes in large protein interaction networks', *BMC Bioinformatics*, Vol. 4, No. 2.

Brandes, U. (2001) 'A faster algorithm for betweenness centrality', *Journal of Mathematical Sociology*, Vol. 25, pp.163–177.

Dunn, R., Dudbridge, F. and Sanderson, C. (2005) 'The use of edge-betweenness clustering to investigate biological function in protein interaction networks', *BMC Bioinformatics*, Vol. 6, No. 39.

Girvan, M. and Newman, M. (2002) 'Community structure in social and biological networks', *PNAS*, Vol. 99, pp.7821–7826.

Kaplan, N., Vaaknin, A. and Linial, M. (2003) 'PANDORA: keyword-based analysis of protein sets by integration of annotation sources', *Nucleic Acids Research*, Vol. 31, pp.5617–5626.

Kelley, B., Sharan, R., Karp, R., Sittler, T., Root, D., Stockwell, B. and Ideker, T. (2003) 'Conserved pathways within bacteria and yeast as revealed by global protein network alignment', *PNAS*, Vol. 100, pp.11394–11399.

Koyuturk, M., Grama, A. and Szpankowski, W. (2005) 'Pairwise local alignment of protein interaction networks guided by models of evolution', *Proceedings of ACM RECOMB*, pp.48–65.

Maslov, S. and Sneppen, K. (2002) 'Specificity and stability in topology of protein networks', *Science*, Vol. 296, pp.910–913.

Newman, M. and Girvan, M. (2004) 'Finding and evaluating community structure in networks', *Physical Review E*, Vol. 69, pp.026113 (15 pages).

Rives, A. and Galitski, T. (2003) 'Modular organization of cellular networks', *PNAS*, Vol. 100, pp.1128–1133.

Sharan, R., Ideker, T., Kelley, B., Shamir, R. and Karp, R. (2004) 'Identification of protein complexes by comparative analysis of yeast and bacterial protein interaction data', *Proceedings of ACM RECOMB*, pp.282–289.

Xenarios, L., Salwinski, L., Duan, X., Higney, P., Kim, S. and Eisenberg, D. (2002) 'DIP, the database of interacting proteins: a research tool for studying cellular networks of protein interactions', *Nucleic Acids Research*, Vol. 30, pp.303–305.