Identification of Functional Modules in Protein Interaction Networks

Lei Shi

Department of Computer Science and Engineering

State University of New York at Buffalo

Protein-Protein Interaction (PPI)

Biological Meaning of PPI

- Proteins interact with each other for stability and functionality
- Most cellular functions are performed in a protein complex level
- Interaction evidence is interpreted as functional coherence / consistency

Determination of PPIs

- Experimental methods

  Yeast two-hybrid systems, Mass spectrometry, Protein microarray
- Computational methods

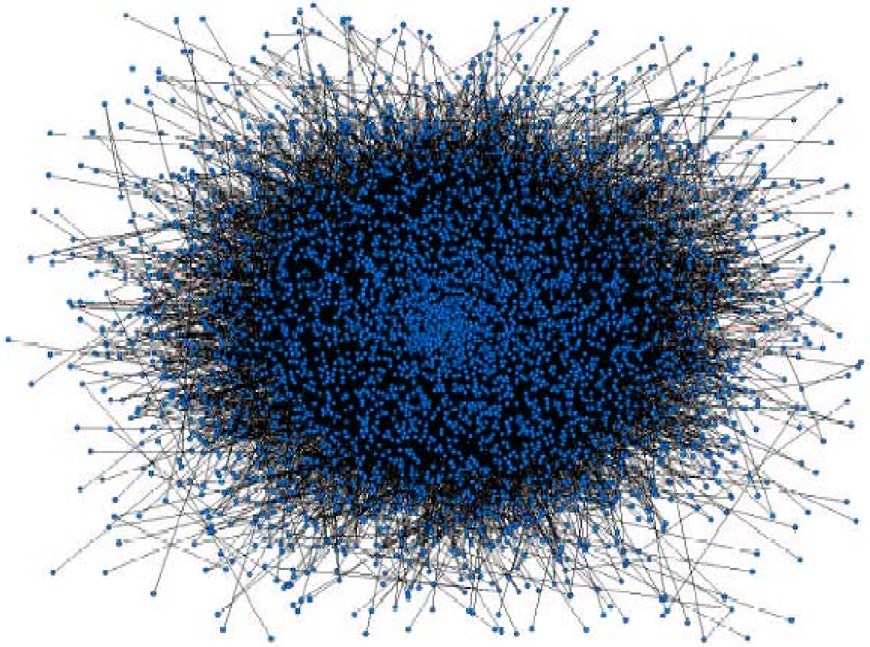
  Homology search, Gene fusion analysis, Phylogenetic profiles

Protein Interaction Network

- **Representation of Protein Interaction Networks**
  - Undirected, un-weighted/weighted graph $G(V,E)$,

    a set of nodes V as proteins and a set of edges E as interactions

- **Problem of Protein Interaction Networks**
  - Large scale
  - Complex connectivity
  - Noisy



Protein-Protein Interaction (PPI)

- **Weighted Network of PPI**
  - Common neighbor based method

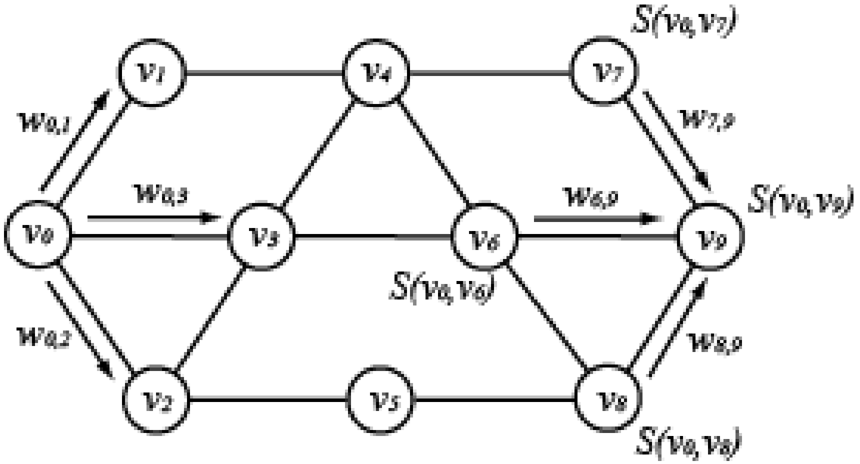
$$W_{(i,j)} = \frac{Com(i,j)*2}{N(i)+N(j)}$$

Functional Influence Model

Functional Influence

- $$S(p) = \lambda \prod_{i=0}^{n-1} \frac{w_{i(i+1)}}{\delta} \cdot \frac{1}{d_i} \qquad \text{where } p = \langle v_0, v_1, \cdots, v_n \rangle$$
  $$= \frac{\lambda \cdot w_{0,1}}{\delta} \prod_{i=1}^{n-1} \frac{w_{i(i+1)}}{\delta} \cdot \frac{1}{d_i} \qquad \text{when } d_0 = 1$$
- Influence factors: normalized weights, inverse of degree

Measurements

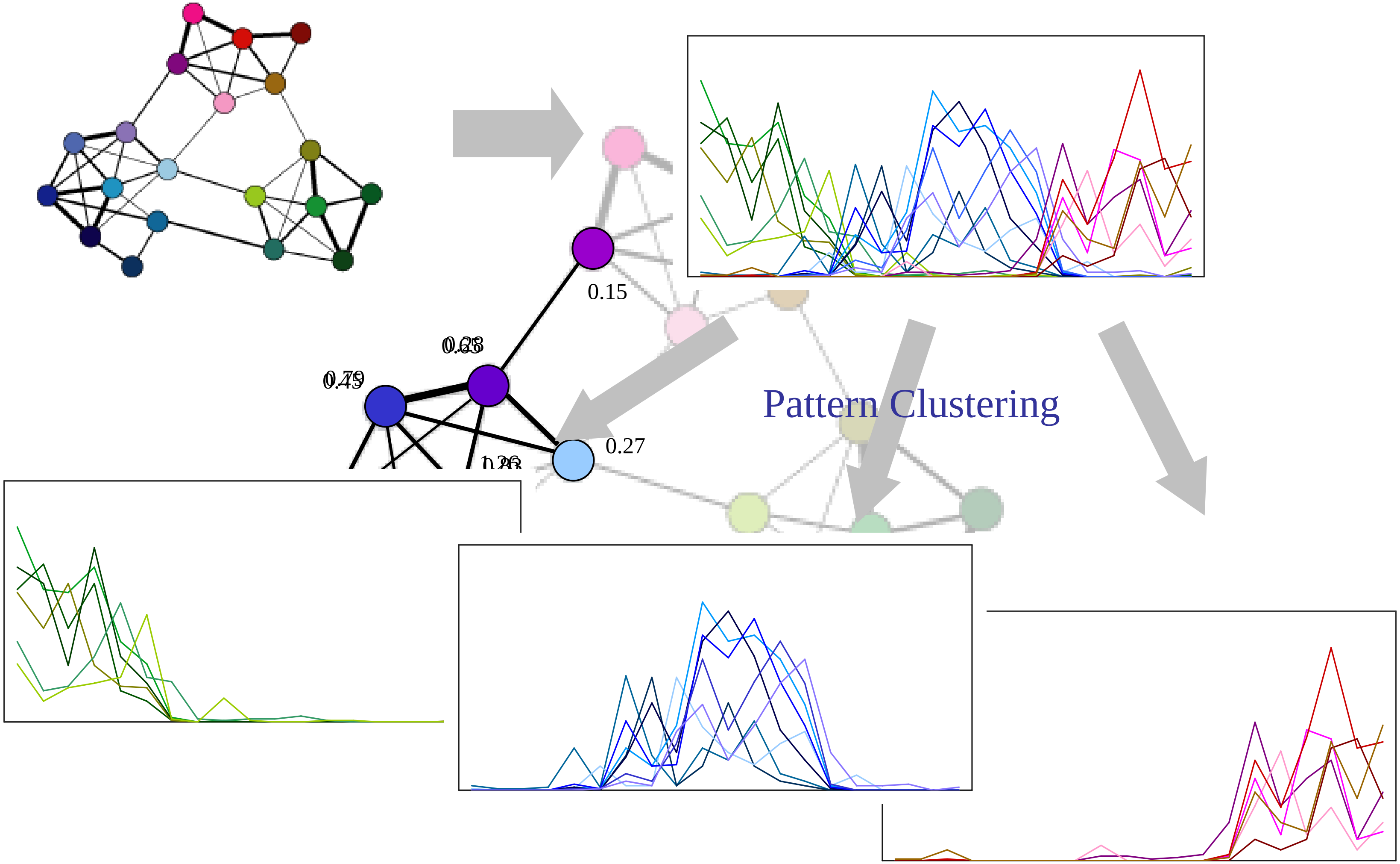


Flow Simulation

- **Algorithm**

1. Initialize $inf_s(s)$
2. Compute initial flow $f_{init}(s \rightarrow y)$ by $\boxed{f_{init}(s \longrightarrow y) = \frac{w_{s,y}}{\delta} \times inf_s(s)}$
3. Update $inf_s(y)$ by $\boxed{inf_s(y) = \sum_{x \in N(y)} f_s(x \longrightarrow y)}$
4. Compute flow $f_s(y \rightarrow z)$ by $\boxed{f_s(y \longrightarrow z) = \frac{w_{y,z}}{\delta} \times \frac{inf_s(y)}{|N(y)|}}$
5. Repeat 3 and 4 until $f_s(y \rightarrow z)$ is less than a threshold θ

Schematic View



Pattern Clustering

Clustering Methods

- **Partitional clustering**

  e.g., restricted neighborhood search, Markov clustering, K-means.

- **Density-Based Clustering**

  e.g., maximum clique, quasi clique, clique percolation

- **Hierarchical Clustering**

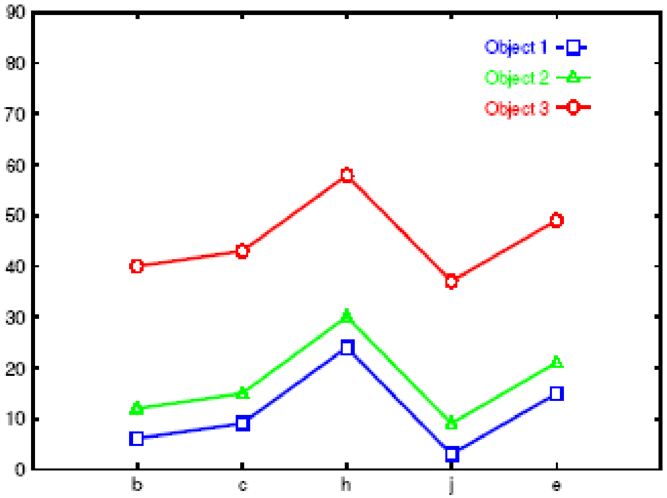
  e.g. Bottom-up approaches, e.g., distance-based, common neighbors

  Top-down approaches, e.g., minimum cut, betweenness cut

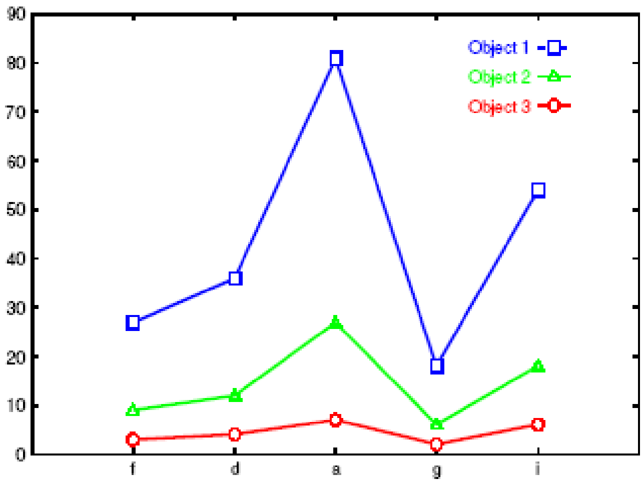
Clustering Methods

- **P-Clustering**

shift



scaling



Clustering by Pattern Similarity in Large Data Sets, by **Haixun Wang**, Wei Wang, Jiong Yang, and Philip S. Yu, in the ACM International Conference on Management of Data (SIGMOD), June 2002

Paralleling Algorithm

- Assign each processor with n/p nodes
- In slave processors, random walk n/p nodes in the graph and output a array as the result of functional flow for each node assigned.
- The master processor will gather the results and do the clustering based on the results.

Paralleling Algorithm

```
int main (int argc, char *agrv[]){
….
        MPI_Status status;
        MPI_Init ( &argc, &argv );
        MPI_Comm_size ( MPI_COMM_WORLD, &nProcs );
        MPI_Comm_rank ( MPI_COMM_WORLD, &id );

        …
        if (id != master){


                    Network * subnw = new Network ("networkData");
                n_lo = (id-1 )* (n / nProcs) + 1;
                n_hi =   id * (n / nProcs);
                …
                // do randomwalk for each node.
                for (i = n_lo; i <= n_hi; i = i + 1) {
                        Char funtionalFlow[] = subnw->randomWalk(i);
                        // send functional flow to the master node
                        MPI_Send (functionalFlow, arraySize, MPI_CHAR, master, tag, MPI_COMM_WORLD);
                }

        }

        if ( id == master){
                     Cluster *cl = new Cluster();
                    for(i = 0;i< n ; i++){
                                    // receive functional flow from each node and insert into map.
                                    MPI_receive(rFunFlow, arraySize, MPI_CHAR, MPI_ANY_SOURCE, tag,MPI_COMM_WORLD, &status );
                                    cl->insert( rFunFlow);

                    }
                    cl->runClustering();
        }
}
```

Experiment Setup

- **Date Source:**

  MIPS protein-protien interaction data

  3882 nodes 13877 interactions

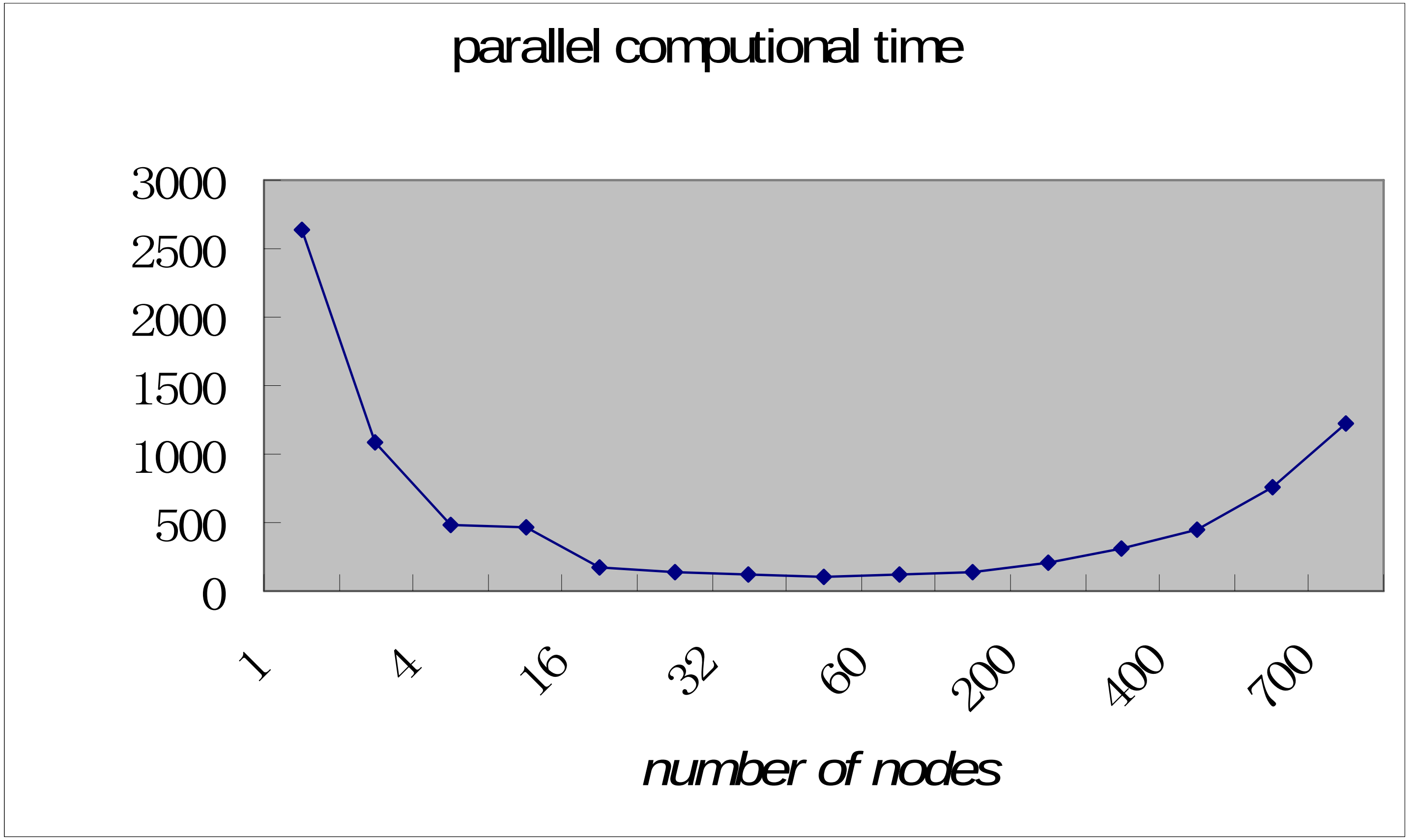
- **Cluster:**

  The u2 cluster which consists of 1056 dual processor DELL SC1425 compute nodes.

- **Computation:**

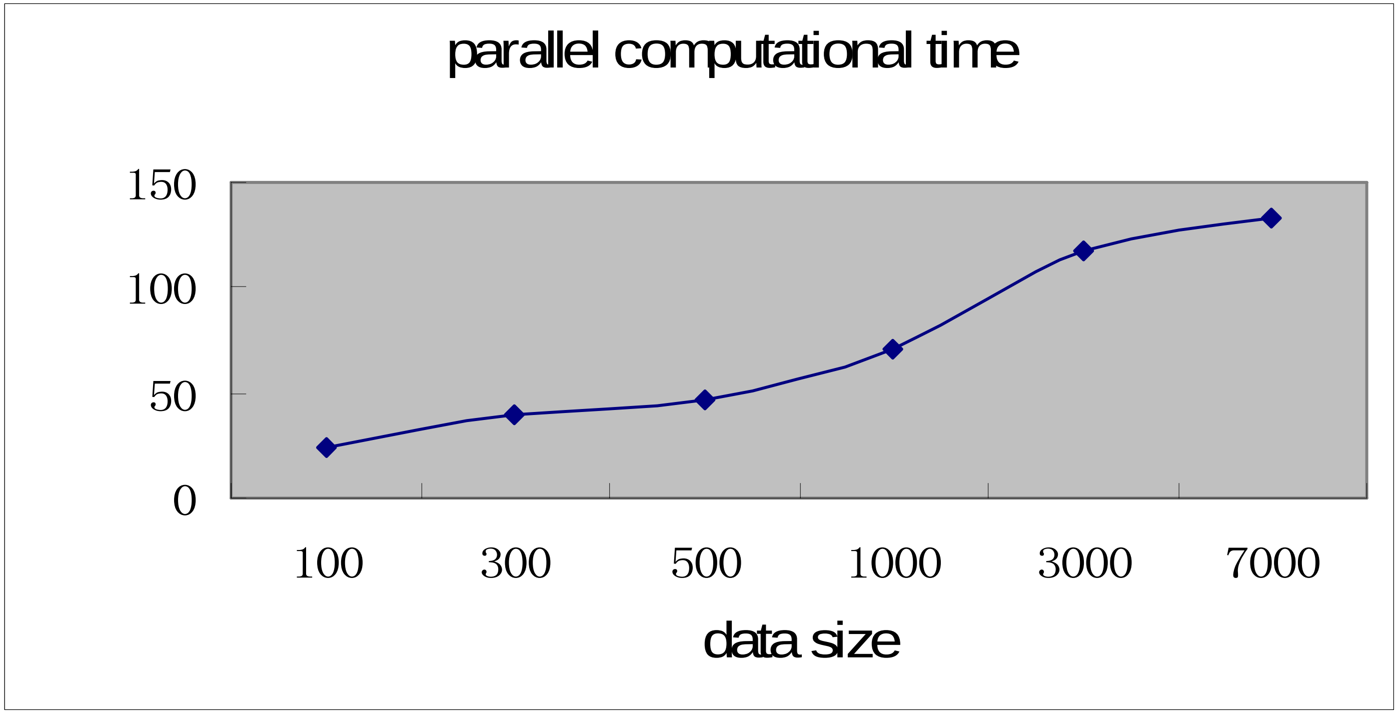
  For each fixed number of nodes, compute 10 times and get the average as the computational time.

Result (time vs. number of nodes)



Date Size: **3882 nodes 13877 interactions**

Result (time vs. data size)



Number of nodes : 32