A Cytoscape plugin for identifying functional modules in biological networks

Denys Kukushkin, Yijie Wang, and Xiaoning Qian

Department of Computer Science and Engineering, University of South Florida

Abstract

The recent advances in biochemical high-throughput technology have enabled the systematic analysis of genome-scale biological networks. As interactions among biomolecules play essential roles in complex behavior of life systems, understanding the functional organization of this tremendous amount of biomolecules is one of critical challenges in modern systems biology. To assist biomedical researchers in analyzing genome-scale life systems, we develop a computational algorithm to identify node clusters as potential functional modules in genome-scale biological networks. The advantage of our computational algorithm is that it aims to identify clusters that are both densely connected and sparsely connected, both of which manifest important interaction patterns for different cellular functionalities. This project focuses on the implementation of the above mentioned algorithm as a functional module identification plugin for an open source network analysis platform called Cytoscape. The significance of this plugin is that it can help biomedical researchers effectively analyze large-scale biological networks of their interest. Given a biological network, our plugin can perform the clustering algorithm to identify groups of biomolecules in the network as potential functional modules for further analyses of their biological significance.

Introduction

This poster will focus on the analysis of protein protein interaction (PPI) networks. Proteins play important roles in a variety of cellular processes including DNA replication, cell cycle control, and signal transduction [1]. Identification of functional modules within a PPI network is important in further understanding of underlying cellular mechanisms. Most of the current algorithms search for densely interacting protein groups. Still, it is possible for a group of proteins to perform a similar function, but have no interaction between each other (Ex. transmembran proteins rarely interact with themselves, but interact with cytoplasmic proteins as well as extra-cellular ligands [2]). The goal of this project was to create a program which would be capable of identifying functional modules within a PPI network, in which proteins can be either densely or sparsely connected.

Methodology

A PPI network can be represented as a graph $G(V, E)$, where V is the set of nodes (corresponding to proteins) and E is the set of edges (corresponding to interaction between two proteins). The algorithm proposed by Wang [3] is capable of identifying both types of protein clusters. The program itself is implemented as a plugin for an open source platform Cytoscape [4]. Such choice allowed to focus more on the plugin implementation, while its visualization of a graph was left to Cytoscape.

Dense clusters

Sparse clusters
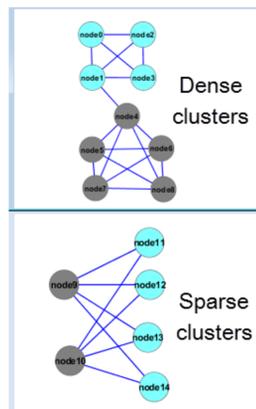
Plugin Functionalities

Cytoscape was written in Java programming language; therefore, our plugin implementation is also Java-based. Due to this, our plugin is platform independent and can be deployed on any device which has Java Virtual Machine (JVM).

Plugin interface
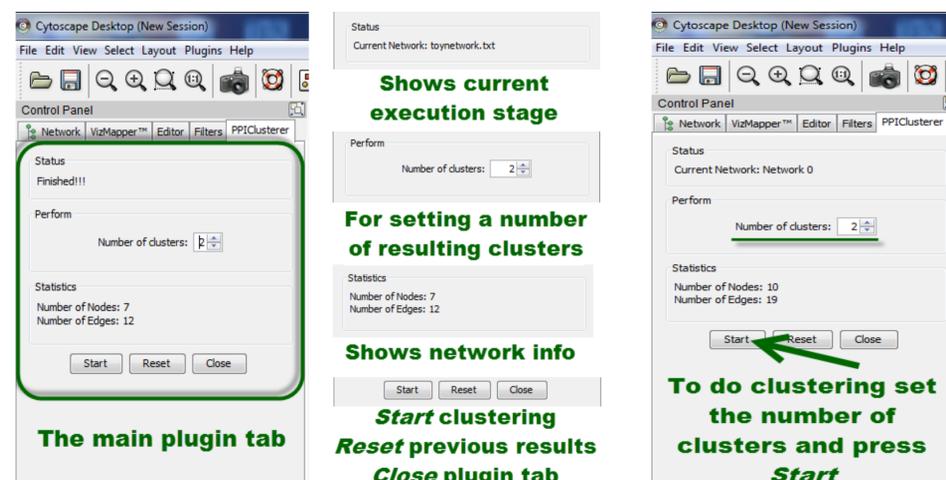
Shows current execution stage

For setting a number of resulting clusters

Shows network info

Start clustering
Reset previous results
Close plugin tab

The main plugin tab

To do clustering set the number of clusters and press *Start*

Plugin Execution

Given a graph $G(V,E)$, the algorithm described in [3] use an adjacency matrix A recording the interactions among proteins, a degree matrix D, and a desired number of clusters k as its input. The algorithm will output a cluster assignment matrix X. For the plugin, the input will be a raw network G without any identified clusters. The output of the plugin will be a processed network G with clusters being identified. Therefore, we get the following execution steps of the plugin:

Step 1: Calculate an adjacency matrix A and a degree matrix D of a given graph G

Step 2: Run Wang's algorithm [3] using results from Step 1 and value k being taken from the *Number of clusters* field

Step 3: Using results from Step 2, assign each node to a corresponding cluster and build a cluster result table

Plugin Execution Results

As a result of plugin's execution, a table with resulting clusters will be created. Nodes within a cluster can be highlighted by clicking on the corresponding cluster's row in the table. Some additional functionalities of the plugin include coloring of nodes in the same cluster with the same color and arranging clusters in a circular manner.

Results Panel	
Results	
Total number of clusters: 9	
Cluster #	Info
Cluster 0	Number of Nodes: 32
Cluster 1	Number of Nodes: 3
Cluster 2	Number of Nodes: 9
Cluster 3	Number of Nodes: 28
Cluster 4	Number of Nodes: 7
Cluster 5	Number of Nodes: 13
Cluster 6	Number of Nodes: 20
Cluster 7	Number of Nodes: 9
Cluster 8	Number of Nodes: 7

Resulted Colored Network

Resulted Circular Layout
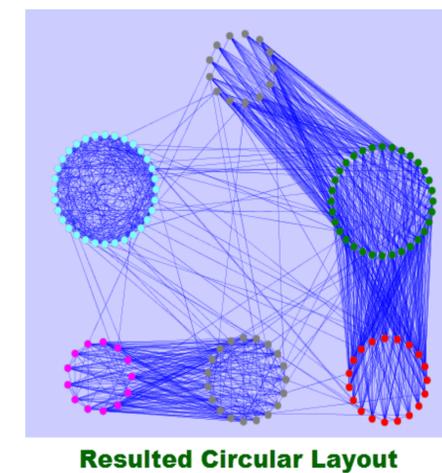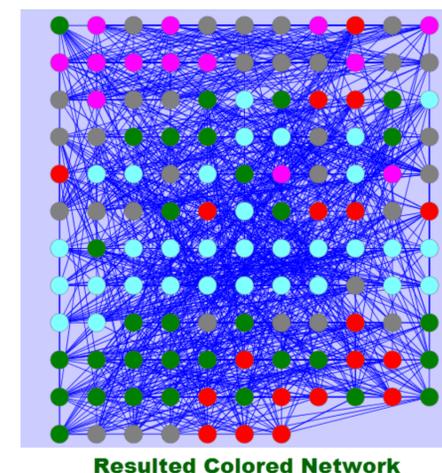
Conclusions

We have designed a plugin which is capable of identifying functional modules within biological networks. The plugin can identify protein clusters in which proteins have similar interaction patterns, either densely or sparsely connected. Identification of these clusters will play an important role in providing an insight into the underlying cellular processes of large-scale networks.

References

[1] Phizicky EM and Fields S. Protein-protein interactions: Methods for detection and analysis. Microbiol Rev, 1995; 59: 94-123.

[2] Pinkert S, Schultz J and Reichardt J. Protein interaction networks: More than mere modules. PLoS ComputBiol, 2010; 6: e1000659.

[3] Wang Y and Qian X. Blockmodel module identification in protein interaction networks through Markov random walk. European Signal Processing Conference 2013, invited paper.

[4] Shannon P, Markiel A, Ozier O, Baliga NS, Wang JT, Ramage D, Amin N, Schwikowski B and Ideker T. Cytoscape: a software environment for integrated models of biomolecular interaction networks. Genome Research, 2003; 13(11): 2498-504.