NetworkAnalysis 1.0: a java-based software for the analysis of undirected networks

Xuan Liu

Abstract: Recent developments of various high-throughput techniques have generated unprecedented quantities of biological data. In parallel with the rapid expansion of omics, researchers have shown a dramatic interest in predicting the organization of complex biological systems by interpretation of those high-throughput data. Thus, we developed a java-based, lightweight software, NetworkAnalysis 1.0, for researchers to analyze biological networks' properties. It provides a user-friendly graphical interface. It allows users to add a new connection to the existed network, and analyze network characteristics, including average degree, degree distribution and hubs in the network with great accuracy.

Availability: NetworkAnalysis 1.0 is implemented in Java.

1 Introduction

Identification of complex interactions between cellular components, such as protein-protein interactions (PPI), metabolic network and protein-RNA interactions, is critical to clarify genetic regulatory systems[1]. To gain the global properties of a complex system, building a network and monitoring its nodes features is a critical method used in the field of system biology[2].

In PPI networks, a node is an abstraction of protein. Edges represents the interaction between two proteins or self-connection of a dimer. Degree is the basic feature of a node, which denotes the number of edges related to the node[3]. It essential to perform a systematic analysis of PPI as it makes great contribution to the understanding of the potential pathways required for emergence of drug resistance[4].

2 Methods

The NetworkAnalysis software was developed in IntelliJ IDEA 2020.3 (https://www.jetbrains.com/idea/) and implemented in Java. It based on two packages: (1) Analyzer package to store entity classes and accomplish main functions of this software (2) sample package for the design of graphical user interface (GUI) based on javaFX and the layout of GUI was developed in SceneBuilder -8.5.0 (https://gluonhq.com/products/scene-builder/).

Entity classes	Node. java
	Edge. java
	Network. java
Network analysis	Operation.java
Utility class	CsvWriter.java
GUI design	Controller.java
	GUI.fxml
	Main.java

Table 1. The program structure of NetworkAnalysis 1.0.

2.1 Class Node

The Node class represents nodes that existed in the PPI network. It contains two variables: (1) *nodeName* variable is a string variable (2) edgeNum variable is an int variable. It is used to simplify the operation, as the degree of each node needs to be calculated in many methods in other classes. The Node class contains two constructors and several methods to get or set the value of two variables. The *equal* method has been overridden in Node class.

2.2 Class Edge

The constructor of Edge class receives two nodes as arguments and makes an edge between them. The *getNodeA* and *getNodeB* methods are applied to return those two nodes. Alike in the Node class, *equal* method has been overridden.

2.3 Class Network

The Network class contains nodeMap, which receives the name of the node as the key. It also involves a list of edge (edgeList) and a default constructor. The *getNodeMap* and *getEdgeList* method returns nodeMap and edgeList, respectively.

The *addNode* and *addEdge* methods are used to add new nodes to nodeMap and add a new connection to edgeList, separately. The *isEdgeExist* method is used to test whether the connection already exists. Those three methods are used simultaneously in the *addInteraction* method in Operation class.

2.4 Class Operation

Operation class comprises methods used for network analysis. For convenience, all methods in this class are defined as *static* methods and can be called without instantiating an object. Operation class clarifies a class variable. The *readNetworkFile* method receive the path of an input file as the parameter. It utilizes BufferedReader to read the input file and HashMap to store data. It is worth noted that the instance object of a network is created in the *readNetworkFile* methods; thus, reading an input file is the first step to conduct network analysis in NetworkAnalysis 1.0. The *checkNetwork* method is used to check whether the network object already declared. It is called in every action in the Controller class.

The *addInteraction* method adds new nodes and interactions to the existed network, and the *getNodeEdgeNum* method conducts the function of searching for the degree of a specific node. The *getAverageDegree* method allows users to calculate the average degree of the network. Hubs, nodes with the highest degree of a network, can be gained by *getHubs* method. The *writeNetworkFile* method generate output files with the help of uses *CsvWriter* class.

The *NetworkStatus* method prints the total node number, interaction or edge numbers, average degree, and the highest degree and hubs in the network on the console. The *printInteractionList* method prints the edited network and monitor changes in the network continuously.

2.5 GUI design

The layout code of graphic interface was created by Scene Builder, which is a UI designer tool worked with javaFX. The FXML file named “GUI.fxml” was generated by Scene Builder. The layout of GUI incorporates three types of elements, including TextField, Button and TextArea. TextFeilds allow users to enter

the name of proteins for use on subsequent analysis. TextArea displays the output information showed on the console. Each layout element was allocated with an action through the use of *javafx.event.ActionEvent* class and corresponding methods were defined in the Controller class. Buttons can trigger the corresponding events. Controller class only creates one instance of the Network class. Consequently, NetworkAnalysis 1.0 only supports analyzing one network at once. The Main class involves the *main* method, which provides an entry point and executes the java application.

3 Results

3.1 Console output

Using protein-protein interaction network (PPInetwork.txt) as an example, we conducted an analysis for its network states.

The graphic interface was shown in Figure 1. Before importing a plain text file, the output region displayed a blank page. After adding a network file to the software, the file path and the initial network status were automatically showen in the output region, as shown in Figure 1. The graphic interface enables users to add new interactions and get the degree of a specific node manually.

The result of the sample PPI network showed there are 296 proteins and 242 edges existed in the network. The average degree of the network is around 1.6351. Protein with UniProt ID P32851 is the hub of the

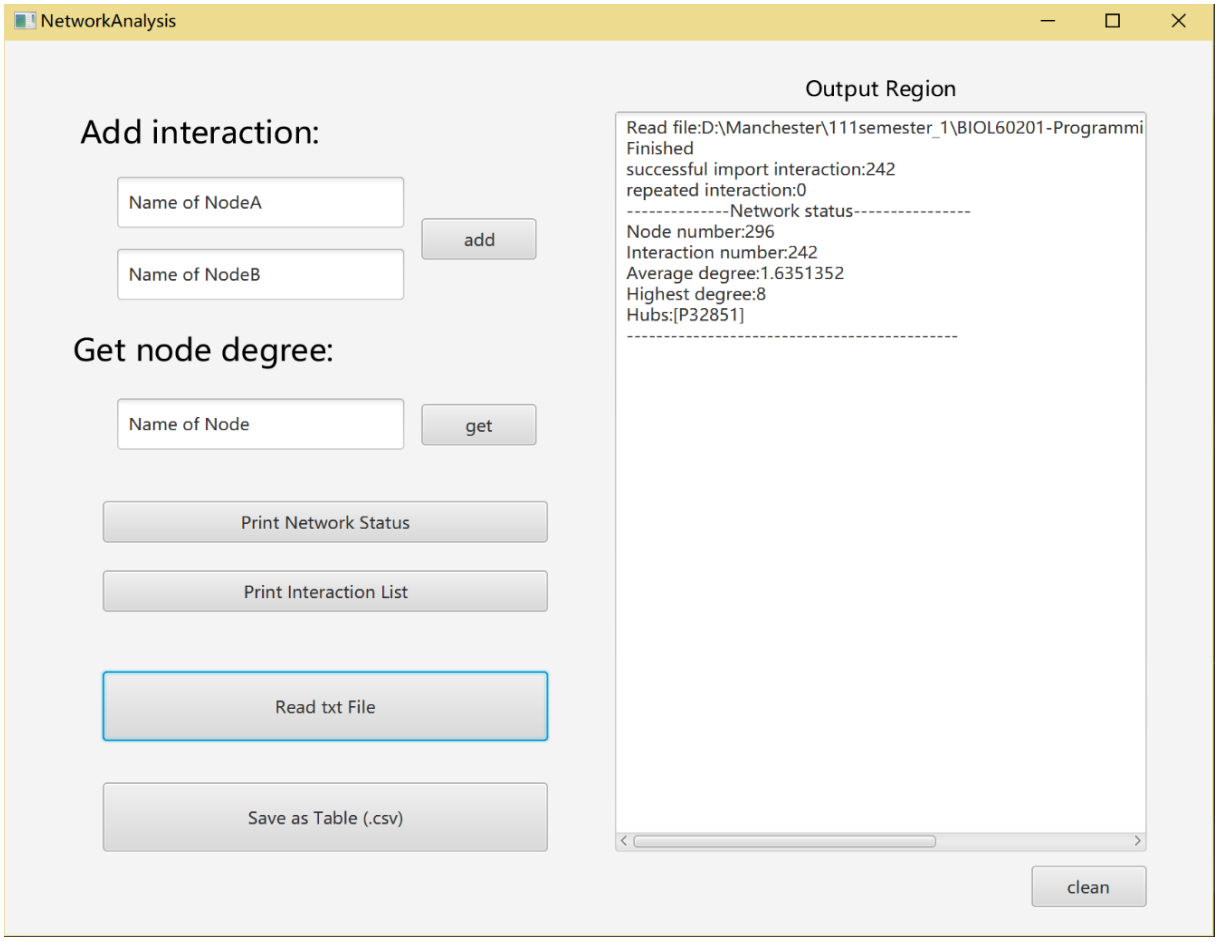

Figure 1. Graphic interface of NetworkAnalysis 1.0

network with degree 8. P32851 is the Syntaxin-1A in rats and is responsible for hormone and neurotransmitter exocytosis and endocytosis[5].

3.2 File output

The example output file was shown in Table 2. Figure 2 used a histogram to describe the principal feature of the sample network, the degree distribution, which revealed how centralized or distributed of a network and the type of a network. It revealed that the sample network is a scale-free network, as most nodes in the network have a low degree, and few nodes, such as P32851, have a high degree. The deletion of hubs is likely to deteriorate information transfer flow in a PPI network[6]. In novel drug development, hubs are mainly used as potential drug targets [7].

Degree	Nodes
1	190
2	58
3	27
4	13
5	5
6	2
7	0
8	1

Table 2. Example output

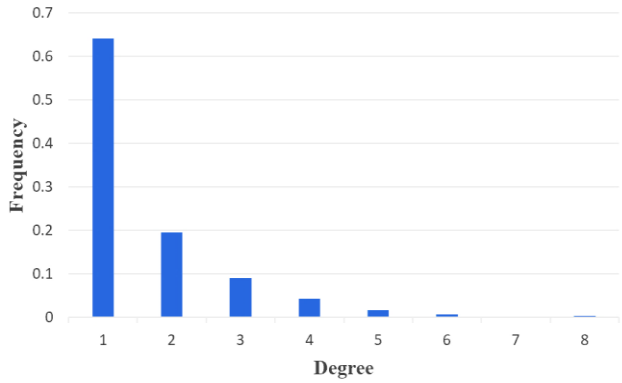

Figure 2. Histogram of degree distribution

4 Discussion and conclusion

We compared the analysis performance of NetworkAnalysis 1.0 with Cytoscape 3.8.2 (https://cytoscape.org/) and String 11.0 (https://string-db.org/)[8, 9]. The defining characteristic of NetworkAnalysis 1.0 is that it is a lightweight software with little demand for computer storage space and provides limited analysis options. When processing the example input file, the PPI network, Cytoscape 3.8.2 proved a network visualization option and occupied about 1063MB of memory, while NetworkAnalysis 1.0 occupied much less than that. Compared to String v11.0, NetworkAnalysis 1.0 is not limited to PPI and can be used in all types of undirected network, while String v11.0 offers an option for gene-set enrichment analysis.

The major weakness of NetworkAnalysis 1.0 is that it cannot offer network visualization, or set combined score and thickness of lines between nodes to illustrate the strength of interactions.

In term of future work, it is useful to expend functionalities of NetworkAnalysis 1.0, such as evaluating the closeness centrality to identify drug resistance pathways.

References

[1] Modeling and Simulation of Genetic Regulatory Systems: A Literature Review. 2002;9(1):67-103.

[2] Klipp E. Systems Biology: A Textbook: Wiley-VCH; 2016.

[3] Barabasi A-L, Oltvai ZNJNrg. Network biology: understanding the cell's functional organization. 2004;5(2):101-13.

[4] Raman K, Chandra N. Mycobacterium tuberculosis interactome analysis unravels potential pathways to drug resistance. BMC Microbiol. 2008;8:234-.

[5] Consortium TU. UniProt: a worldwide hub of protein knowledge. Nucleic Acids Research. 2018;47(D1):D506-D15.

[6] Csermely P, Korcsmáros T, Kiss HJ, London G, Nussinov RJP, therapeutics. Structure and dynamics of molecular networks: a novel paradigm of drug discovery: a comprehensive review. 2013;138(3):333-408.

[7] Raman K, Vashisht R, Chandra NJMB. Strategies for efficient disruption of metabolism in Mycobacterium tuberculosis from network analysis. 2009;5(12):1740-51.

[8] Li M, Li D, Tang Y, Wu F, Wang JJIjoms. CytoCluster: a cytoscape plugin for cluster analysis and visualization of biological networks. 2017;18(9):1880.

[9] Szklarczyk D, Gable AL, Lyon D, Junge A, Wyder S, Huerta-Cepas J, et al. STRING v11: protein–protein association networks with increased coverage, supporting functional discovery in genome-wide experimental datasets. 2019;47(D1):D607-D13.