

OPEN ACCESS

Full open access to this and thousands of other papers at <http://www.la-press.com>.

SBEToolbox: A Matlab Toolbox for Biological Network Analysis

Kranti Konganti¹, Gang Wang², Ence Yang² and James J. Cai^{1,2}

¹Whole Systems Genomics Initiative, Texas A&M University, College Station, Texas 77843-4458, USA. ²Department of Veterinary Integrative Biosciences, Texas A&M University, College Station, Texas 77843-4458, USA.
Corresponding author email: jcai@tamu.edu

Abstract: We present SBEToolbox (Systems Biology and Evolution Toolbox), an open-source Matlab toolbox for biological network analysis. It takes a network file as input, calculates a variety of centralities and topological metrics, clusters nodes into modules, and displays the network using different graph layout algorithms. Straightforward implementation and the inclusion of high-level functions allow the functionality to be easily extended or tailored through developing custom plugins. SBEGUI, a menu-driven graphical user interface (GUI) of SBEToolbox, enables easy access to various network and graph algorithms for programmers and non-programmers alike. All source code and sample data are freely available at <https://github.com/biocoder/SBEToolbox/releases>.

Keywords: Matlab toolbox, biological network, node centrality, network evolution

Evolutionary Bioinformatics 2013:9 355–362

doi: [10.4137/EBO.S12012](https://doi.org/10.4137/EBO.S12012)

This article is available from <http://www.la-press.com>.

© the author(s), publisher and licensee Libertas Academica Ltd.

This is an open access article published under the Creative Commons CC-BY-NC 3.0 license.

Introduction

The complexity of biological systems represents an enormous intellectual challenge to researchers who mostly remain reliant on correlative approaches to biology. The volumes of biological network data gathered in systems biology have outpaced their ability to assimilate them into real knowledge and the requirement of software tools for analyzing network data is continually growing. Network analysis software usually transforms the network data into a graph framework in order to take distinct advantages of being able to adopt techniques developed in graph theory, engineering, and computer science. In our opinion, good network analysis software should include a sufficient number of graph algorithms being efficiently implemented as functions, which are flexible enough to be extended easily, and accessible through easy-to-use user interfaces (UIs). With such software, it is possible for users without programming expertise to directly relate specific biological interactions with the network properties and dynamics.

In the following sections, we describe how we designed Systems Biology and Evolution Toolbox (SBEToolbox), subsequently discuss various algorithms implemented for network analysis, and finally mention the advantages as well as potential drawbacks of our software and point out our future research roadmap.

Implementation

SBEToolbox is implemented using a combination of native Matlab code and MEX/C functions based on the Boost Graph Library (Fig. 1).¹ Functions developed in native code can be classified into: (1) core functions that execute the tasks organized under SBEToolbox's menu; and (2) helper functions that automate repetitive tasks and assist core functions. SBEToolbox can read and write network information in three commonly used network file formats: tab-delimited, SIF, and Pajek. It saves the network

information on disk in a Matlab MAT-file for each working session as an $n \times n$ sparse adjacency matrix representing the network of n nodes. The adjacency matrix is loaded into the variable named *sbeG*, and the node information is stored in a cell string vector *sbeNode*. SBEGUI is a figure window to which user-operated controls are organized as a drop-down menu to access core functions of the SBEToolbox (Fig. 2). All major functions can be accessed from the menu and it is easy and simple for the end-users to load a network file, compute statistic measures for the network and its nodes, detect highly connected node clusters (or modules) using graph clustering algorithms, and evolve the networks. Different graph layout algorithms were either implemented natively (Random, Circle, and Tree Ring) or incorporated from Matlab BGL (Kamada-Kawai Spring, Gürsoy Atun, and Fruchterman-Reingold) so that small- to medium-sized networks can be plotted and manipulated as standard figures using Matlab's built-in figure controls. Additionally, graphs can be exported to external network analysis tools such as Cytoscape² and Pajek³ for further analysis, and visualization libraries such as Protovis and Sigmajs for further illustration. Since network information is written to disk and network variables remain unchanged between sessions, SBEToolbox is highly customizable through the use of SBE-plugins. New functions can be developed from built in template code as SBE-plugins. We define SBE-plugins as custom functions, which can easily access the network information for current working session that will work on the command line, and can also be easily incorporated into SBEGUI by using built-in plugin management tools. Links to screencasts on: (1) how to install SBEToolbox; (2) how to detect, visualize and export the network modules, as well as links to WIKI pages describing SBE-plugin development are provided in the README file with the software download.

Results

Our main strategy for developing this new network analysis tool is to use Matlab, which has been one of the default choices of programming language for efficiently dealing with matrix manipulations. Matlab itself is a “merging” language that can seamlessly integrate with many other languages such as

SBEGUI		SBE-plugins	
SBEToolbox			
MatlabBGL	Native matlab code	Third-party applications	
Boost graph library			

Figure 1. Structure and organization of SBEToolbox and related software components.

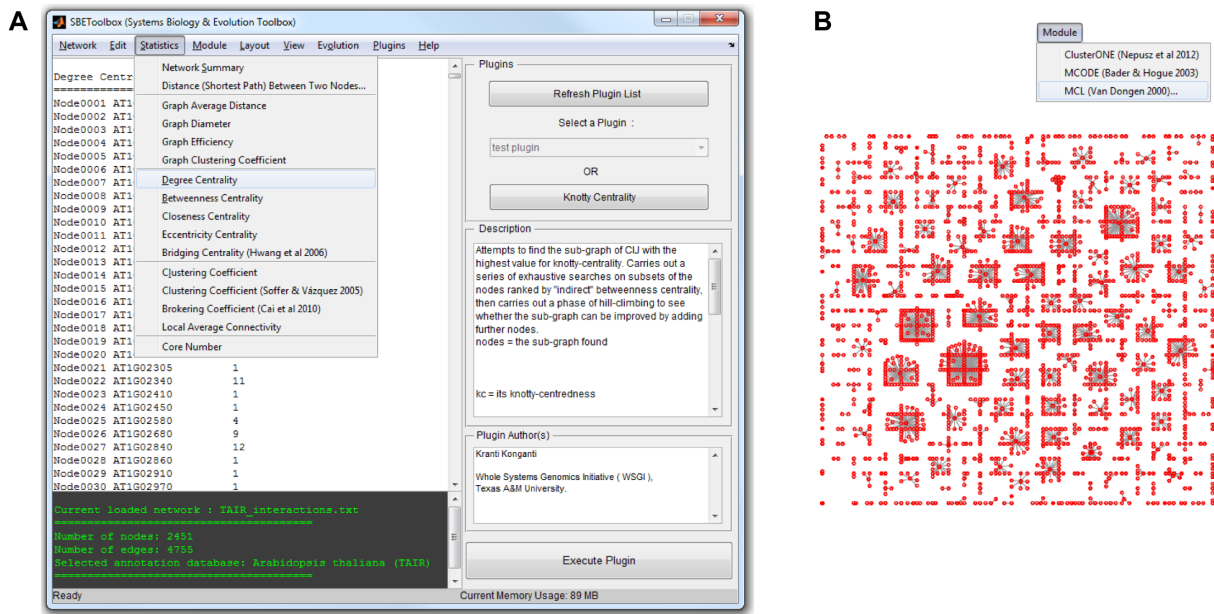


Figure 2. SBEGUI—the main interface of SBEToolbox (left) and an overview visualization of detected network modules (ie, clusters of highly connected nodes) (right). The modules were detected by using the MCL algorithm.

C and Java, and functions written in Matlab have features that stay on the continuum between low- and high-level languages. In addition, Matlab has a sophisticated plotting library, and the software tools developed in Matlab inherit these distinct characteristics; however, to our knowledge, a Matlab toolbox specifically designed for biologists with graphical user interfaces to conduct comprehensive network analyses is still missing. SBEToolbox intends to fill this gap.

Main functions

SBEToolbox covers a wide range of algorithms for computing network statistics. These algorithms include commonly used ones, such as betweenness centrality, clustering coefficient, and closeness centrality, as well as newly developed ones, such as bridging centrality,⁴ Soffer's clustering coefficient,⁵ and brokering coefficient.⁶ Statistics that can be computed using the SBEToolbox also include local average connectivity, core number, graph mean distance, graph diameter, graph efficiency, current information flow,⁷ neighborhood connectivity,⁸ participation coefficient,⁹ rich-club coefficient,¹⁰ and so on. Random networks can be generated using Erdős-Rényi, small-world, and ring lattice algorithms. SBEToolbox's module-detecting functions clusters nodes into highly connected subnetworks or modules using three differ-

ent algorithms: MCODE;¹¹ ClusterOne;¹² and MCL.¹³ MCODE is based on vertex weighting by local neighborhood density and outward traversal from a locally dense seed node to isolate the dense regions, whereas MCL is based on the simulation of stochastic flow in a graph. ClusterOne generates overlapping clusters and has been shown to outperform MCODE and MCL in predicting members in protein complexes.¹² Our toolbox includes all three algorithms to facilitate the users who intend to compare predictions with different algorithms. Cluster membership information of nodes is displayed in the output window and detected modules can be plotted.

SBEToolbox contains a collection of general-purpose functions that can facilitate applications of specific methods to real datasets. For example, input and output functions allow file conversion between different formats, and plot functions allow network visualization using interfaces to external programs such as Cytoscape² and Pajek³ (<http://pajek.imfm.si>). It also uses JavaScript libraries, Protovis (<http://mbostock.github.com/protovis>), and Sigma.js (<http://sigma.js.org>) to render scalable vector graphic (SVG) plots that can be displayed in web browsers. These third-party applications are sandboxed with the software to maintain integrity and minimize failure. All functions are linearly implemented to solve a particular task, and the links between functions are through input

and output variables. This simple design allows users to extend the functionality, as well as to implement and integrate their own functions into the software as SBE-plugins more easily. For example, we developed a plugin for a network motif detection tool, mfinder using built-in plugin management tools. Helper functions of the software that communicate to different external programs can be customized and reused to incorporate other external programs according to each user's necessity.

Network evolution

Using SBEToolbox, users can simulate the evolution of a network as a stochastic process involving node duplication, node loss, and edge rewiring (Fig. 3). The users control the simulated evolutionary process with three parameters: (1) the number of generations, g , which is the number of nonoverlapping steps for which the simulation should run; (2) evolutionary rate, r , which is the rate of node duplication (or node loss), where r is the total number of nodes – or in case of edge rewiring, it is the total number of edges; and (3) fixation probability, p_{fix} , which is the likelihood of a designated evolutionary event (for example, node duplication, node loss, or edge rewiring) becoming fixed per generation.

Feature comparison between network analysis toolboxes

Many valuable software tools have been developed both within and outside the discipline of systems biology. Some of the similar toolboxes to SBEToolbox that have been developed in Matlab are: Functional Genomics Assistant (FUGA),¹⁴ Brain Connectivity Toolbox (BCT),¹⁵ and Mathworks Bioinformatics Toolbox (MBT). The current version of MBT has a few basic graph theory algorithms, but it does not have functions for any kind of statistical analysis. BCT and FUGA have a good number of statistical analysis functions, and the latter infers network through expression analysis and provides annotation. However, both of these toolboxes lack a unified solution for integrating graphic user interfaces, network evolution, and the ability for users to prototype and share custom functions through plugin distribution. A major feature comparison between these toolboxes is given in Table 1. Several non-Matlab-based tools also exist for network analysis and visualization. We believe that Matlab's built-in functions allow for rapid prototyping of new algorithms, and its efficient handling of data manipulation characteristics can be easily leveraged and extended using SBEToolbox. A feature comparison between SBEToolbox and other

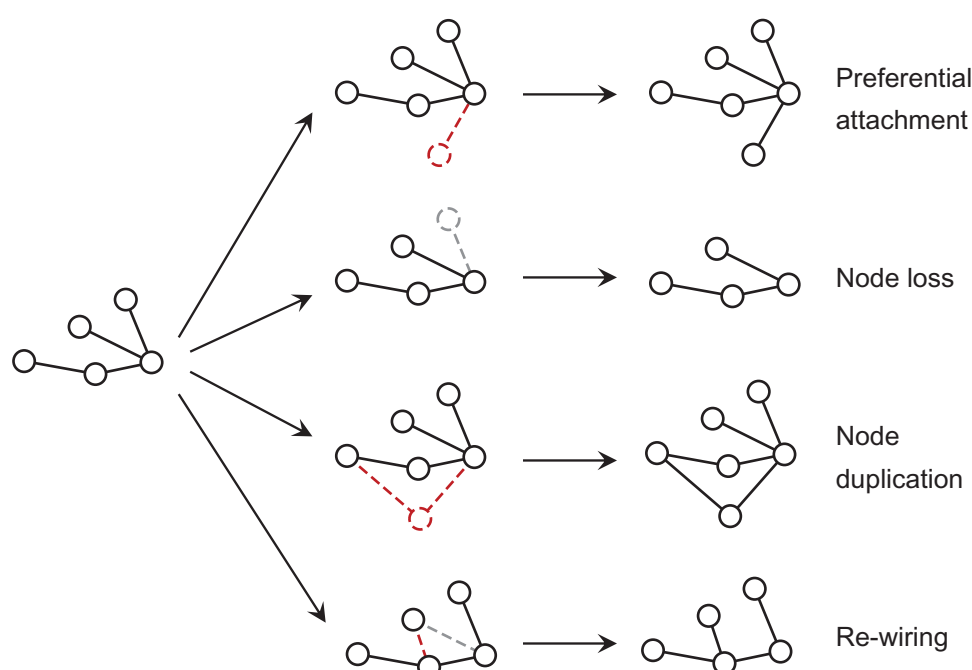


Figure 3. Four modes of network evolution implemented in the SBEToolbox: (1) preferential attachment; (2) node loss; (3) node duplication; and (4) edge-rewiring.

Table 1. Feature comparison between SBEToolbox and relevant Matlab-based toolboxes FUGA (Functional Genomics Assistant), BCT (Brain Connectivity Toolbox), and MBT (Mathworks Bioinformatics Toolbox).

	SBEToolbox	FUGA	BCT	MBT
Centrality calculation	✓	✓	✓	×
Module detection	✓	✓	×	×
Node (gene) annotation	✓	✓	×	×
Network evolution	✓	×	×	×
GUI	✓	×	×	×
Programmable plugins	✓	×	×	×
Plugin management tools	✓	×	×	×

non-Matlab-based network analysis tools is provided in Supplementary Table 1.

Scalability and performance of SBEToolbox

Efficient implementation of SBEToolbox's functions results in minimal usage of memory and disk space when working with a network file, which allows users to handle moderately large-scale network data on a standard desktop computer. To analyze a global human physical protein interaction network¹⁶ containing about 10,000 nodes and more than 80,000 edges, about 850 MB of memory was required and network file stored for the working session occupied about 300 bytes of disk space; conversely, for a random Erdős-Rényi network containing 10,000 nodes and more than 450,000 edges, about 2 GB memory was required to complete all the analyses, while the actual network file stored on the disk for the session occupied approximately 1 MB of disk space. For both the networks, all of the core functions finished in less than 10 minutes. For even larger networks, the parallel computing ability of Matlab can be easily leveraged to solve the problems caused by the high requirement of computational resources.

We set out to compare SBEToolbox with similar Matlab toolboxes mentioned in Table 1 in terms of scalability and performance. Since these toolboxes have different profiles, a couple of common and important network topological metrics, betweenness centrality and clustering coefficient, were chosen to test memory usage and computation times on net-

works of varying node and edge sizes. MBT was excluded from the test as it does not have functions for computing the considered statistics. We noticed that SBEToolbox, FUGA, and BCT used similar amounts of memory for computing these two statistics, which varied approximately from a minimum of 40 MB to a maximum of 200 MB. With the help of built-in Matlab time functions, we saw a major difference in computation times between these toolboxes. In both these tests, SBEToolbox's computation time was much faster than FUGA and BCT (Supplementary Fig. 1). BCT was not even able to finish computing betweenness centrality for a small network of about 1,000 nodes in reasonable time. All of the analyses and tests were run on a Macintosh OS X (10.7 Lion) laptop computer with 4 GB of RAM and a 1.7 GHz Intel 64-bit processor.

Numerical validation of native Matlab code for the MCL algorithm

Taking advantage of built-in matrix functions available in Matlab, we were able to implement the MCL algorithm¹³ natively in less than 50 lines of code. To validate our implementation, we compared the clustering results obtained by native code with those obtained by using the mcl program (based on C) available in the MCL-edge software (<http://micans.org/mcl/index.html>). The comparison was performed with a network of 330 nodes obtained from the study of Ideker et al.¹⁷ (This dataset is available in the example_dataset folder provided with SBEToolbox as a .sif format file, galFiltered_330_nodes.sif). First, the MCL-edge source code was downloaded and compiled on a Mac OSX with i64 architecture. The mclload binary was used to convert the .sif format file to .mci format file to run mcl with default options, which resulted in a total of 97 clusters. Next, MCL was executed from SBEGUI, which also found exactly 97 clusters with a minor difference. The difference in the number of n -node clusters identified between the two versions of MCL algorithm implementation is indicated in Supplementary Table 2. Two implementations produced nearly identical results: SBEToolbox's mcl.m resulted in 38 two-node clusters and 28 three-node clusters, whereas MCL-edge's mcl resulted in 39 two-node clusters and 27 two-node clusters. SBEToolbox's mcl.m found an extra three-node cluster consisting of



nodes YER079W, YKL204W, and YNL154C. Node YER079W is also shared with another three-node cluster, which contains nodes YER079W, YHR135C, and YNL116W, which was also identified by MCL-edge's mcl (Supplementary Fig. 2). The two-node cluster (YNL154C, YKL204W) identified by MCL-edge's mcl was absent from SBEToolbox's mcl.m results. Apart from this minor difference, the number of n -node clusters and all the nodes participating in each cluster were exactly identified in both cases.

Example application of SBEToolbox in characterizing disease genes

Genes that underlie human inherited diseases are important subjects in systems biology research. We have previously demonstrated in detail, using SBEToolbox in analyzing the human protein–protein interaction data,¹⁶ that in order to reveal important network properties of disease genes, providing new insights to the origin and etiology of disease is necessary.⁶ Here, we briefly summarize the key points of our discoveries. We introduced a new statistical measure named the brokering coefficient and used this statistic to discern between disease and nondisease genes based on their distinct network properties. The brokering coefficient is a composite metric (ie, for each node, it is calculated as $\log(d) - \log(c)$, which is the difference between the log-transformed degree, d , and the clustering coefficient, c). In a network, a node (or gene) with a large brokering coefficient tends to have more neighbor nodes, while the number of connections between these neighbor nodes themselves tends to be small. Based on our analysis, disease genes have unusually higher degrees and lower clustering coefficients (ie, larger brokering coefficient) than nondisease genes.⁶ Thus, disease genes are more likely to be broker genes in networks, in that they connect many other proteins that would not be connected otherwise.

Availability and System Requirements

All versions of SBEToolbox can be freely downloaded from <https://github.com/biocoder/SBEToolbox/releases>. Users can submit bugs and follow the development cycle of our toolbox at <https://github.com/biocoder/SBEToolbox/issues>. The minimum requirements for the software are:

- **Matlab:** The SBEToolbox has been developed in Matlab version R2012b and makes use of all the improvements made to the core Matlab. Although, the codebase works in previous versions as well, some new features may be incompatible.
- **Disk space:** Approximately 200 MB of disk space is needed for installation, most of which is due to sandboxed third-party applications and annotation databases.
- **Memory:** We recommend a minimum of 4 GB of random-access memory for faster computations, although this is not mandatory.
- **Central processing unit:** 1.5 GHz processor or better.
- **Windows XP or newer, Mac OS X 10.6 or newer with i64 architecture, Linux.**

Conclusion

SBEToolbox is flexible, easy-to-use, and highly customizable from our point of view, and it provides researchers with an interactive tool to explore biological networks, as well as to compute centralities and topological statistics for the networks. The output of a function is displayed in the output window and can be saved as a file, copied to clipboard, or exported as a variable to the Matlab workspace. The extensive plotting ability of Matlab allows users to create publication-quality plots. We strongly believe that the extensibility of the software through a standardized SBE-plugin protocol will increase the functionality of the toolbox and will be a great resource for the systems biology research community using the Matlab system to develop new algorithms and generate hypotheses from network datasets. While currently SBEToolbox only supports undirected networks, in an ongoing effort, we plan to add support for weighted networks.

Acknowledgments

We thank TAMU Whole Systems Genomics Initiative for providing systems administration support and hosting internal source code repository.

Author Contributions

Conceived and designed the draft software and UI: JJC. Developed the software and user interface: KK.

Contributed code to the software: EY, GW. Wrote the manuscript: KK, JJC. Agree with manuscript results and conclusions: EY, GW. All authors reviewed and approved of the final manuscript.

Funding

Author(s) disclose no funding sources.

Competing Interests

Author(s) disclose no potential conflicts of interest.

Disclosures and Ethics

As a requirement of publication the authors have provided signed confirmation of their compliance with ethical and legal obligations including but not limited to compliance with ICMJE authorship and competing interests guidelines, that the article is neither under consideration for publication nor published elsewhere, of their compliance with legal and ethical guidelines concerning human and animal research participants (if applicable), and that permission has been obtained for reproduction of any copyrighted material. This article was subject to blind, independent, expert peer review. The reviewers reported no competing interests.

References

1. Gleich D. MatlabBGL—A Matlab Graph Library [webpage on the Internet]. David Gleich; 2008. Available from: <http://dgleich.github.com/matlab-bgl/>. Accessed 2008.
2. Shannon P, Markiel A, Ozier O, et al. Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome Res*. 2003;13(11):2498–504.
3. de Nooy W, Mrvar A, Batagelj V. *Exploratory Social Network Analysis with Pajek (Structural Analysis in the Social Sciences)*. Revised and Expanded Second Edition. Cambridge, UK: Cambridge University Press; 2011.
4. Hwang W, Cho Y, Zhang A, Ramanathan M. *Bridging Centrality: Identifying Bridging Nodes in Scale-free Networks*. Technical Report 2006-05. Buffalo, NY: Department of Computer Science and Engineering, University at Buffalo; 2006. Technical Report 2006-5.
5. Soffer SN, Vázquez A. Network clustering coefficient without degree-correlation biases. *Phys Rev E Stat Nonlin Soft Matter Phys*. 2005;71(5 Pt 2):057101.
6. Cai JJ, Borenstein E, Petrov DA. Broker genes in human disease. *Genome Biol Evol*. 2010;2:815–25.
7. Missiuro PV, Liu K, Zou L, et al. Information flow analysis of interactome networks. *PLoS Comput Biol*. 2009;5(4):e1000350.
8. Maslov S, Sneppen K. Specificity and stability in topology of protein networks. *Science*. 2002;296(5569):910–3.
9. Guimerà R, Nunes Amaral LA. Functional cartography of complex metabolic networks. *Nature*. 2005;433(7028):895–900.
10. Colizza V, Flammini A, Serrano MA, Vespignani A. Detecting rich-club ordering in complex networks. *Nat Phys*. 2006;2:110–5.
11. Bader GD, Hogue CW. An automated method for finding molecular complexes in large protein interaction networks. *BMC Bioinformatics*. 2003;4:2.
12. Nepusz T, Yu H, Paccanaro A. Detecting overlapping protein complexes in protein-protein interaction networks. *Nat Methods*. 2012;9(5):471–2.
13. Van Dongen S. *Graph Clustering by Flow Simulation* [dissertation]. Utrecht, The Netherlands: University of Utrecht; 2000.
14. Drozdov I, Ouzounis CA, Shah AM, Tsoka S. Functional Genomics Assistant (FUGA): a toolbox for the analysis of complex biological networks. *BMC Res Notes*. 2011;4:462.
15. Rubinov M, Sporns O. Complex network measures of brain connectivity: uses and interpretations. *Neuroimage*. 2010;52:1059–69.
16. Bossi A, Lehner B. Tissue specificity and the human protein interaction network. *Mol Syst Biol*. 2009;5:260.
17. Ideker T, Thorsson V, Ranish JA, et al. Integrated genomic and proteomic analyses of a systematically perturbed metabolic network. *Science*. 2001;292:929–34.



Supplementary Materials

Supplementary Data: [SupplementaryData.pdf](#).

Supplementary Table 1: Feature comparison between SBEToolbox and non-Matlab-based network analysis software.

Supplementary Table 2: Comparison of MCL results obtained by using native Matlab code and MCL-edge program.

Supplementary Figure 1: Comparison of computation times between SBEToolbox, BCT and FUGA for betweenness centrality and clustering coefficient.

Supplementary Figure 2: Demonstration of different clustering of 5 nodes in SBEToolbox's mcl.m versus MCL-edge.