Methods for Random Modularization of Biological Networks

Zachary M. Saul* and Vladimir Filkov†
Department of Computer Science
University of California, Davis
Davis, CA 95616
*Contact Author: saul@cs.ucdavis.edu
†filkov@cs.ucdavis.edu

Abstract— **Biological networks are formalized summaries of our knowledge about interactions among biological system components, like genes, proteins, or metabolites. From their global topology and organization one can learn nontrivial, systemic properties of organisms. In studies of biological network organization empirical networks are typically compared to random network models, and features are identified as important if they are statistically "unusual," i.e. occur surprisingly often or seldom. Naturally, more representative random models result in better feature identification. Since biological networks exhibit a modular structure (mostly pertaining to their hierarchical functional organization), random network models need be modular similarly.**

In this work we consider the problem of generating random network models that incorporate network modularity. Theoretically, the problem is equivalent to generating random decompositions of a graph into a given number of connected components. Here we describe two methods we have developed to do that and illustrate their utility on pertinent systems biology problems of feature scaling.

I. Introduction

Biological networks provide convenient summaries of our knowledge about interactions among components in biological systems (e.g. genes and proteins). When represented as mathematical graphs with nodes (components) and edges (interactions) between the nodes, they can be visualized and analyzed on whole-system scales. Then, questions of structure and organization can be posed and answered, and hypotheses relating the topological structure to biological phenomena can be investigated.

In systems biology, networks are analyzed by comparing them to random networks or simulated networks, and features are identified as important if they are statistically "unusual" (i.e. occur surprisingly often or seldom). Statistically, this amounts to eliminating the null hypothesis with very high confidence. To find out exactly what "usual" is, the biological network is typically modeled as a random graph having as many properties of the original as possible. Naturally, better random models of what is usual in these networks will yield better discriminatory power of the "unusual" features. Thus, global properties of the networks are very important and are used to create the random graph distributions on which feature identification relies.

Recently much attention has been focused on the statistical distribution of the node degrees in biological networks (i.e. the number of interactions per node). The evidence seems to point to the presence of many more highly connected nodes (or hubs) in the networks than one would expect, yielding a *scale-free* topology. The highly connected nodes tend to have important biological roles (e.g. lethal genes transcriptional regulators, etc.) [1], [2], [3]. It is relatively simple to generate random networks with node degree distributions mimicking those of real biological networks [1], [4]. Scale-free random graphs fare better than Erös-Rényi random graphs in describing reality [1], and have been used successfully to identify systemic properties of networks [5].

Another key observation about biological networks is that they have a modular structure [6], [7]. Studying modularity has a number of important precedents outside of biology. For example, in both circuit and software design modularization is considered a *simplifying* design practice [8]. Studying naturally occurring biological modularity allows us to take advantage of system analysis techniques borrowed from these other fields.

There are varied examples of biological network modularity, but most pertain to the hierarchical organization of function in the networks. Biochemical pathways are an obvious example of functional modularity in metabolic networks [9]. In addition, a number of researchers have published evidence that within protein-protein interaction networks there are sets of nodes that act as modules, interacting mainly with other nodes in the module [3], [10]. Others have performed studies that indicate that transcriptional regulation occurs in a modular fashion [11]. There, it has been shown that sets of transcription factors (TF) act together to activate (or inhibit) genes, and each TF set (or module) regulates its gene in a unique spatial or temporal context.

Another study noted that there are subgraphs in both naturally occurring and man-made networks which occur more often than expected due to chance [5]. The authors proposed that these subgraphs could be elementary modular building blocks. Studying such over-represented subgraphs, or *network motifs*, could aid understanding of the system properties of biological networks.

To build better random network models, in addition to simulating properties like node degree distribution, one needs to factor in network modularity. Then, one can attempt to answer systemic questions like: if a property has been shown

Sixth IEEE Symposium on BionInformatics and BioEngineering (BIBE'06)
0-7695-2727-2/06 $20.00 © 2006 IEEE

to hold in the whole network, will it also hold in the network's parts? In other words, one can answer if a given property *scales* to different organizational levels of the network and reason as to why it does or why it does not.

The simplest random graph model that can be used to further such studies is one which has the same number of modules or components as the real graph, but they have been selected at random. Such a graph could serve as a null hypothesis for testing if properties are independent of the level of modular organization of the biological network.

In this work, we consider the problem of generating random modularizations of biological networks. Theoretically, the problem is equivalent to generating random decompositions of a graph into a given number of connected components. Here, we describe two methods that we have developed to do that and illustrate their utility on pertinent systems biology problems of feature scaling.

Our methods can be used to simulate the distribution of modules expected due to chance in a given graph. This distribution gives a baseline context in which to analyze any modular network. In the context of network module analysis, a distribution of the modules is analogous to the distribution of graphs in the context of more traditional single-node analysis described above. In both of these contexts, scientists can use the simulated distribution to calculate z-scores for network features of interest, assigning a score of potential importance to each. This is an important property, because it gives a statistical indication of which features are "surprising" or important.

The algorithms community has explored a number of methods to decompose a graph into connected subgraphs [12], [13], [14]; however, these methods investigate graph decomposition as a way to increase the speed of graph algorithms, rather than focusing on *random* decompositions. As far as we know no standalone algorithms are available to randomly decompose a graph into connected components, for the purpose of simulating a distribution.

In addition to the biological motivations for a method to randomly decompose a graph into modules, there exist a few additional benefits of such a method. First, from an algorithmic standpoint, modularizing networks leads to a natural divide and conquer strategy. Many of the algorithms currently used to analyze biological networks are computationally complex. In fact, most of them are not executable in polynomial time (in the size of the network). To exacerbate this problem, biological networks often consist of thousands of interactions, exceeding our computing capacity. Breaking biological networks into modules and running an algorithm on the modules independently can improve performance.

Next, from an analysis standpoint, studying modules can can reduce the number of factors a researcher must consider when studying a biological network, making the analysis more tractable than it might be. Because these networks consist of the thousands of interactions, wholesale system analysis is often beyond the capacity of our analysis tools. By modularizing the network, the researcher can consider the parts of the network independently. This use of system modularization is common in the engineering world, where systems are often composed of several interacting modules.

II. METHODS

Given these motivations, our purpose is to randomly decompose the input graph into n connected components (modules). Both methods that we developed first select n seed nodes at random and then grow the components from these seed nodes in a round-robin fashion, selecting one node to add to each module per round. The round robin process can be likened to choosing teams in a pick-up sports match. The modules take turns "choosing" eligible nodes to add (in this case, the modules can only add neighboring nodes that haven't already been chosen). Clearly, with this method of "module-growing," it is possible to attain a module for which there no longer exist any nodes eligible to be added. At this point, our algorithm marks the module as inactive and no longer considers it in the round-robin process. This round-robin process applies to both of our algorithms, and they differ only in the way that they select nodes to be added.

Algorithm 1 The BFS node-selection algorithm picks nodes in breadth first order to be added to the current module.

INPUTS:
CurModule - Module for which to select a node.
UsedSet - Set of previously used nodes.
ActiveModules - Set of active modules.

```
Node = CurModule.NeighborQ.Remove ()
while Node ∈ UsedSet do
  if CurModule.NeighborQ.Empty () then
    ActiveModules.Remove (CurModule)
    Output (CurModule)
  end if
  Node = CurModule.NeighborQ.Remove ()
end while
UsedSet.Add (Node)
CurModule.NodeSet.Add (Node)
CurModule.NeighborQ.Add (Neighbors of Node)
return (CurModule, UsedSet, ActiveModules)
```

A. BFS algorithm

The first algorithm uses a node-selection method that is patterned after breadth-first search.

The breadth-first search algorithm maintains a queue of nodes to be visited in the graph. The queue is initialized with the neighbors of the start node for the search. Then, the nodes are visited in the order that they are in the queue, and as each is visited, its neighbors are added to the back of the queue. This procedure guarantees a breadth first search.

Similarly, the BFS node-selection method maintains a neighbor-queue for each active module. This queue contains the neighbors of the module. Each time a node is added to the module, its neighbors are added to the back of the queue,

guaranteeing that the nodes are added to the module in breadth first order. However, because we have the restriction that each node can be added to only one module, our algorithm keeps track of which nodes have already been used and checks before adding a node to a module if that nodes has been previously used.

B. Random-BFS algorithm

The second node-selection method is also patterned after breadth-first search. However, instead of choosing the neighbors in visited order, this method chooses the nodes randomly from the set of neighbors of the module and is called random-BFS. Rather than storing the candidate nodes in queues, random-BFS simply keeps a set of neighboring nodes for each module, and it selects each node for addition uniformly at random from the set of neighbors for each module.

Algorithm 2 The random-BFS algorithm is similar to the BFS algorithm with the exception that it selects nodes from the neighbor set uniformly at random rather than in visited order.

INPUTS:
CurModule - Module for which to select a node.
UsedSet - Set of previously used nodes.
ActiveModules - Set of active modules.

```
Node = CurModule.Neighbors.RandomRemove ()
while Node ∈ UsedSet do
  if CurModule.Neighbors.Empty () then
    ActiveModules.Remove (CurModule)
    Output (CurModule)
  end if
  Node = CurModule.Neighbors.RandomRemove ()
end while
UsedSet.Add (Node)
CurModule.NodeSet.Add (Node)
CurModule.Neighbors.Add (Neighbors of Node)
return (CurModule, UsedSet, ActiveModules)
```

C. Overlapping modules and target sizes

Thus far, we have limited our decomposition algorithm to always produce distinct modules. However, in several realistic applications of this method, including biology and engineering, this may not be a reasonable limitation. For example, the modules in a biological pathway graph must overlap. This is how interaction between modules is performed; the shared nodes act as intermediaries between the modules.

To extend our two basic methods to allow for overlapping modules, we have added two parameters. The first parameter is the overlap parameter. If the overlap parameter is set to true, then the test (in both algorithms) to see if a candidate node is already colored is changed to test if that candidate node is colored *the color of the module under consideration*. This technique prevents the same node from being added to a module twice, while still allowing the modules found to contain identical nodes (to overlap). However, because there is no longer a stopping condition when the overlap parameter is enabled, the sizes of the modules desired must be specified with an additional parameter called the target-size vector.

The target-size vector, T, is the second parameter. This is an n-dimensional vector containing target-sizes for each of the n modules. This parameter can be used independently of the overlap parameter. However, if we don't allow overlap, the requested sizes cannot necessarily be obtained, because a graph cannot be divided into connected components with arbitrary sizes. Therefore, in this case T is used to build a vector of "emit probabilities," e. The entries of e are defined as $e[x] = T[x] / \sum_{i=1}^{n} T[i]$ for all x from 1 to n. When a module x comes up in the round-robin process, we first choose whether or not to add a node to it in such a way that we add a node with probability $e[x]$. If we choose to add a node, we select it as before. However, if we choose not to add a node, we simply move to module $x + 1$ in the round-robin process.

III. Results

Biochemists have traditionally organized their studies of large biochemical networks by focusing on the properties of their functionally coherent sub-networks, which are often important to the network as a whole. The large-scale network can then be studied as a collection of the modules, or sub-networks. Likewise, computational studies of modularity in large-scale networks can reveal the relation between local features, or properties, in network modules versus those in the whole network, i.e. property scaling.

Two biologically motivated questions come to mind: (1) are the same local features important in functional sub-networks, or pathways, as in the full network? and (2) does the specific modularization into sub-networks matter?

To illustrate how these questions can be addressed computationally, here we use the fairly simple and popular concept of network motifs as our model for local features. Network motifs are small, 3- or 4-node connected subgraphs which occur suprisingly seldom (or often) in an empirical network, as compared to a random network. Computationally, network motifs are found by counting the occurrence of all possible subgraphs of given number of nodes in the network in question and comparing that count to the one of the occurrence of the same subgraph in a simulated (or random) network in which the node degree distribution is kept constant.

First, we retrieved all forty-eight available biochemical pathways from *E. coli* from the KEGG database [15]. These pathways are models for biochemical functions and are invaluable as examples of biological modules. They range in size from 6 to 38 nodes. Connected together, through common nodes, they comprise a full network of 503 nodes. Then, for each of the pathways, or sub-networks, and for the full network we identified the most-surprising 3-node network motifs (based on their z-score as compared to the expected number based on a same node degree random network). Overall, the motifs that were most common in the sub-networks were also the motifs in the overall network. This relationship is shown in figure 1. The existence of such a scale-invariant property implies that

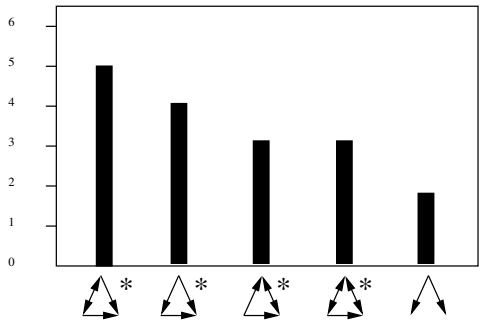

Fig. 1. The number of pathways (out of 48) that each subgraph scored in the top 20% of all motifs. The starred subgraphs were motifs in the full network. Four out of five motifs are motifs at both scales.

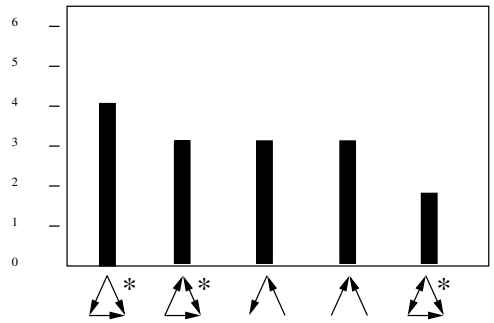

Fig. 2. When the experiment was repeated using random modules, three of the five motifs were the same at both scales.

the process of modularization can leave important network properties intact.

Now we turn to the second question, which provides an excellent example of the utility of the random modularization algorithms: is the scale-invariance of network motifs above a property of the modularization given by the biochemists, or is it a general property that holds in most (or all) biochemical pathways? Using our methods for random network modularization, we repeated the above experiments, with the biochemically derived modules replaced by our randomly generated modules. The results, shown in Fig. 2, are similar to those in Fig. 1. Thus, in spite of the random modularization, the network motifs were mostly independent of scale. This indicates that the scale-independence is a property of the *E. coli* biochemical pathway under study and not a property of the particular modules derived by the biologists.

IV. Conclusions

The methods for random modularization presented here allow us to generate a distribution of modules for a given network. We presented two main methods, each with two parameters. The usefulness of each case depends on the context graph decomposition is desired. For example, in most biological contexts, the existing modularization include overlapping modules. This means that the overlap parameter should usually be used in these situations. Another example is the reasoning behind using the random-BFS method to decompose the graph in the example study. We made this choice because biological pathways generally consist of a compact core with non-compact edge. This is the type of random modularization given by random-BFS. The BFS method wouldn't have suited our purposes because it simply produces compact modules.

The systemic explorations of the *E. coli* example network demonstrate the utility of our methods. In addition, we were able to establish an interesting property of network motifs, their scale-independence, by showing that the prevalence of the top motifs is similar in a functionally and randomly modularized network.

Importantly, we were able to perform our analysis of the behavior of the network at different scales without relying on any assumptions about the underlying degree distribution in our network. This is an important distinction because it is not clear that the commonly made assumption of scale-independence is correct [16].

In general, random modularization facilitates creating background distributions for any biological graph, providing an important tool to study network modularity.

References

[1] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, pp. 509–512, 1999.

[2] H. Jeong, B. Tombor, R. Albert, Z. N. Oltvai, and A.-L. Barabási, "The large-scale organization of metabolic networks," *Nature*, vol. 407, p. 651, 2000.

[3] S.-H. Yook, Z. N. Oltvai, and A.-L. Barabási, "Functional and topological characterization of protein interaction networks," *Proteomics*, vol. 4, pp. 928–942, 2004.

[4] C. Gkantsidis, M. Mihail, and E. Zegura, "The markov chain simulation method for generating connected power law random graphs," in *Proceedings of the 5th Workshop on Algorithm Engineering and Experiments (ALENEX).*, January 2003.

[5] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon, "Network motifs: Simple building blocks of complex networks," *Science*, vol. 298, pp. 824–827, 2002.

[6] L. H. Hartwell, J. J. Hopfield, S. Leibler, and A. W. Murray, "From molecualr to modular cell biology," *Nature*, vol. 402, pp. C47–C52, 1999.

[7] D. A. Lauffenburger, "Cell signaling pathways as control modules: Complexity for simplicity?" *PNAS*, vol. 97, pp. 5031–5033, 2000.

[8] C. Y. Baldwin and K. B. Clark, *Design Rules: The Power of Modularity Volume 1*. Cambridge, MA, USA: MIT Press, 1999.

[9] P. Holme, M. Huss, and H. Jeong, "Subnetwork hierarchies of biochemical pathways," *Bioinformatics*, vol. 19, no. 4, pp. 532–538, 2003.

[10] J.-D. J. Han, N. Bertin, T. Hao, D. S. Goldberg, G. F. Berriz, L. V. Zhang, D. Dupuy, A. J. M. Walhout, M. E. Cusick, F. P. Roth, and M. Vidal, "Evidence for dynamically organized modularity in the yeast protein-protein interaction network," *Nature*, vol. 430, pp. 88–93, 2004.

[11] E. H. Davidson, *Genomic Regulatory Systems*. Elsevier Science and Technology Books, 2001.

[12] J. Fouquet, M. Habib, F. de MontGolfier, and J. Vanherpe, "bimodular decomposition of bipartite graphs," 30th International Workshop on Graph-Theoretic Concepts in Computer Science, WG 2004, 21-23 June 2004.

[13] S. Toida, "A decomposition of a graph into dense subgraphs," *IEEE Transactions on Circuits and Systems*, vol. CAS-32, no. 6, pp. 583–589, 1985.

[14] A. Brandstädt, V. B. Le, and J. P. Spinrad, *Graph Classes: A survey*. SIAM, 1999, ch. 12.

[15] M. Kanehisa, S. Goto, S. Kawashima, and A. Nakaya, "The kegg databases at genomenet," *Nucleic Acids Res*, vol. 30, pp. 42–46, 2002.

[16] N. Pržulj, D. G. Corneil, and I. Jurisica, "Modeling interactome: scale-free or geometric?" *Bioinformatics*, vol. 20, no. 18, pp. 3508–3515, 2004.