Genes Genet. Syst. (2009) **84**, p. 385–395

MODA: An efficient algorithm for network motif discovery in biological networks

Saeed Omidi[1], Falk Schreiber[2,3] and Ali Masoudi-Nejad[1*]

[1]Laboratory of Systems Biology and Bioinformatics (LBB), Institute of Biochemistry and Biophysics and Center of Excellence in Biomathematics, University of Tehran, Iran
[2]Institute of Computer Science, Martin Luther University Halle-Wittenberg, Halle, Germany
[3]Leibniz Institute of Plant Genetics and Crop Plant Research (IPK), Gatersleben, Germany

(Received 20 August 2009, accepted 4 December 2009)

In recent years, interest has been growing in the study of complex networks. Since Erdös and Rényi (1960) proposed their random graph model about 50 years ago, many researchers have investigated and shaped this field. Many indicators have been proposed to assess the global features of networks. Recently, an active research area has developed in studying local features named motifs as the building blocks of networks. Unfortunately, network motif discovery is a computationally hard problem and finding rather large motifs (larger than 8 nodes) by means of current algorithms is impractical as it demands too much computational effort. In this paper, we present a new algorithm (MODA) that incorporates techniques such as a pattern growth approach for extracting larger motifs efficiently. We have tested our algorithm and found it able to identify larger motifs with more than 8 nodes more efficiently than most of the current state-of-the-art motif discovery algorithms. While most of the algorithms rely on induced subgraphs as motifs of the networks, MODA is able to extract both induced and non-induced subgraphs simultaneously. The MODA source code is freely available at: http://LBB.ut.ac.ir/Download/LBBsoft/MODA/

Key words: real-world complex networks, network motifs, subgraph isomorphism, induced and non-induced subgraphs, subgraph sampling, pattern growth approach

INTRODUCTION

In the last few years there has been great interest in the study of complex networks including the World Wide Web (Faloutsos et al., 1999; Bornholdt and Schuster, 2003), biological networks (Middendorf et al., 2005), metabolic pathways (Ouzounis and Karp, 2000; Kanehisa et al., 2008; Masoudi-Nejad et al., 2007), food webs (Dunne et al., 2002) and many other kind of networks. Properties such as small-world effect (Amaral et al., 2000; Watts and Strogatz, 1998), long tailed decay or power-law degree distribution (Barabási and Albert, 1999), clustering coefficient, or degree correlation (Vázquez, 2002), that characterize the large-scale features of complex networks have been identified and widely investigated. Many real-world networks obey a long-tailed degree sequence, often described as a power-law distribution $P(k) \sim ck^{-\gamma}$, where γ is called the scaling exponent of the power-law distribution. Networks that follow the power-law degree distributions are called scale-free (Amaral et al., 2000).

One of the most interesting areas in investigation of complex networks is exploring local structures of networks and finding mutual relations of these local to global properties (Vázquez et al., 2004; Mangan and Alon, 2003). There are basic local interaction patterns or network modules that recurring through different kinds of networks more often than in randomized versions of such networks. These patterns are known as *network motifs* and were first proposed as simple building blocks of complex networks by Milo et al. (2002). Network motifs were originally defined as statistically significant over-represented patterns of local interconnections in complex networks (Alon, 2007). In the case of biological networks, these structures have often been shaped during evolution.

Various aspects of network motifs have been analyzed so far. Vázquez et al. (2004) showed that the global net-

Edited by Takashi Endo
* Corresponding author. E-mail: amasoudin@ibb.ut.ac.ir

work features, for instance clustering coefficient, influence local features such as the abundance of certain subgraphs. Itzkovitz et al. (2003) also demonstrated that the topology of subgraphs affect the value of scaling exponent in scale-free networks via the subgraph maximal degree. In this manner, long-tailed decay of degree distribution indicates the existence of high degree nodes or *hubs* in scale-free networks, and it has empirically shown that many subgraphs aggregate around hubs (Vázquez et al., 2004). Therefore, a network's large scale features and local structure mutually define and predetermine each other. As a result, for an empirical study of theories about large scale features and local structures such as network motifs, it is highly desirable to have efficient tools.

Network motifs have a range of applications: classifying networks into superfamilies (Middendorf et al., 2005), determining the most appropriate network model and verifying parsimony models of phylogeny (Przytycka, 2006), and bringing about new insights in gene regulation (Tsang et al., 2007) are just some cases. However, all of these applications have been handicapped by the small size of motifs of up to eight nodes (Baskerville and Paczuski, 2006). In fact, finding motifs larger than eight nodes is very difficult with existing tools and the size limitation leaves many fundamental questions unanswered. Some of these problems are: Do motifs appear independently, or do they combine to form larger organized structures? How do these network motifs combine to form the global structure of the networks? Are collections of nodes that participate in motifs of larger sizes also more likely to be related to function and/or be conserved through evolutionary history?

Despite of the fact that the existing algorithms for detecting network motifs are not scalable enough to find large network motifs, it would be valuable to design efficient tools for discovering motifs in arbitrary networks. The underlying problem of enumerating subgraph appearances within a network is computationally hard. With growing subgraph size, the time and memory for determining isomorphic subgraphs increase in an exponential manner and the number of non-isomorphic candidate motifs increases exponentially with the size of the motifs (Inokuchi et al., 2000; Kuramochi and Karypis, 2004). Secondly, it is possible to find multiple redundant mappings from a symmetric query subgraph's nodes to the same network's nodes (Grochow and Kellis, 2007); as a result, one subgraph may be overcounted in a network several times which wastes computational resources like CPU time. Thirdly, motifs have to be computed not only in the network itself but also in a sufficient number of randomized networks with some identical properties to the original network to yield a statistically meaningful result, which imposes the extra computational cost for evaluating subgraphs in each random network. Finally, real world networks are often too large (around thousands nodes or even more) and hence dealing with these huge networks, as well as a number of huge random networks of the same size, makes the task more difficult. In consideration of the above challenges, designing an algorithm to handle this diversity of problems is a formidable task.

Here we present a new algorithm for network motif discovery using pattern growth approach called MODA (network MOtif Discovery Algorithm). MODA can compute both induced and non-induced frequent subgraphs as network motifs. The algorithm was implemented in C#. Net. We have tested and compared our algorithm to existing state-of-the-art motif discovery algorithms (Grochow and Kellis, 2007; Milo et al., 2002; Schreiber and Schwöbbermeyer, 2005a; Wernicke, 2006) and can show here the advantages of this approach. In this article, we aim to briefly elucidate the main idea and design principles of the algorithm and therefore will not go into details such as the Null-model, which has been particularly discussed previously.

Previous work The first algorithm for network motif discovery was an exhaustive enumeration of subgraphs in a network (Milo et al., 2002), counting all appearances of all subgraphs of a given size. This approach, however, was not usable with subgraphs larger than five nodes. Another method was introduced by Grochow and Kellis (2007), incorporating an algebraic technique that prevents overcounting for a given subgraph. Their method called *symmetry-breaking condition* can be useful especially for large symmetric motifs, because some kinds of graphs such as stars have many symmetries and computational cost increases drastically (for a star graph with n nodes there are $(n-1)!$ symmetries). For large subgraphs, exhaustive enumerating needs a huge amount of time. On the other hand, when a particular subgraph is exhaustively enumerated we obtain the exact number of occurrences of that subgraph in the network.

Kashtan et al. (2004) introduced a sampling approach to extract network motifs faster than exhaustive search algorithms. Computational time of this method is asymptotically independent of network size. Although their algorithm leads to finding larger motifs in reasonable time, it suffers from *biased* sampling; consequently, it is unable to predict the exact number of motifs impartially.

Wernicke (2006) proposed a very fast algorithm called *FANMOD*, which can be used to detect network motifs up to a size of eight nodes in directed and undirected networks. This algorithm overcomes Kashtan et al. (2004) sampling method's shortcoming through a new method for sampling that leads to *unbiased* sampling. Despite of the fact that this algorithm only extracts induced subgraphs in the network, it disregards non-induced ones. Another tool for network analysis such as network motif discovery and network visualization is MAVisto (Schreiber and Schwöbbermeyer, 2005a), which implements the FPF algorithm (Schreiber and Schwöbbermeyer, 2005b). A recently introduced algorithm that is able to calculate exactly subgraph appearances of a given size is Kavosh (Razaghi Moghadam Kashani et al., 2009). Kavosh finds every subgraph of size k which a particular node participates in. After that, it removes that node and continues by selecting another node until all of the nodes in the network are removed. Kavosh uses a novel technique for counting subgraph appearances that is based on discrete mathematics approach. Data mining methods, which could be used for network motif discovery, are *Apriori-based* and *pattern growth* approaches. Apriori-based methods were originally used for frequent pattern mining in a data mining context. Frequent item set mining or market basket prediction was proposed by Agrawal and Srikant (1994) in a transactional database. The idea could be used for mining frequent subgraphs in a database or collection of graphs. Motif discovery and frequent subgraph mining have common challenges. Subgraph isomorphism as an *NP*-complete problem is the most important of these challenges as the computation cost for subgraph isomorphism grows exponentially with increasing size of the subgraph. In the frequent subgraph mining problem, the frequency of a subgraph is determined by the number of global graphs in the collection of graphs that the subgraph appears in, regardless of whether the subgraph appears many times within a particular graph. Although, this problem differs from network motif discovery and is computationally easier than motif discovery, both problems are related.

Pattern growth methods are based on an extension mechanism such as adding a new edge to a graph; the extension mechanism refers to a general term of extending a pattern by size one that may lead to a new larger pattern. *SPIN* (Huan et al., 2004) is a pattern growth algorithm for mining maximal frequent subgraphs by generating the frequent substructures hierarchically in two steps: starting from trees, and then extending frequent trees to graphs edge by edge. On the other hand, Apriori-based methods (Agrawal and Srikant, 1994) rely on a join operation that joins two (or more) patterns to generate a new pattern. Chen et al. (2006) exploited the *Apriori* property for network motif discovery (see also Ciriello and Guerra, 2008). However, their method has some shortcomings; for example, generating cousins is ambiguous and it is possible to find redundant subgraphs such as several isomorphic subgraphs. Their algorithm is named *NeMoFinder*.

Here we present a new algorithm for network motif discovery that utilizes the pattern growth approach by starting from k-size trees (the simplest form of a connected subgraph) and then extending these trees step by step until a complete graph with k nodes is built. The pattern growth approach helps us to systematically reduce a great number of subgraph isomorphism instances during running time. In order to increase the efficiency of our algorithm, we also exploit sampling throughout the network through a probability distribution that is derived from the degree distribution of the original network's nodes, which increases the speed of our algorithm more than ten times. We also use symmetry-breaking conditions to avoid the overcounting of subgraphs.

MATERIALS AND METODS

Definitions In this paper, $G = (V, E)$ indicates an *undirected graph* where V is a finite set of nodes (vertices), $|V| = n$ is the *graph size* and E is the edge set of the graph which satisfies $E \subseteq V \times V$. An edge $e = \langle u, v\rangle \in E$ is *incident* to nodes u and v and we call u and v *adjacent* nodes. Corresponding *adjacency matrix* A of G is a $n \times n$ matrix, with each entry $a_{i,j} = 1$ if vertices v_i and v_j are adjacent; otherwise $a_{i,j} = 0$. Each node $v \in V$ has *degree* k_v that indicates the number of edges that are incident to v. A *path* between two nodes $u, v \in V$ is a sequence of nodes such that each node in the sequence is adjacent to its immediately next and previous nodes in the sequence and the sequence is started by u and ended by v. A graph is called *connected* if there is a path between any pair of nodes $u, v \in V$. A connected graph with n nodes is a *tree* if and only if it has $n - 1$ edges. If all nodes of G are pairwise adjacent, then G is called a *complete* graph. A complete graph with n nodes has $n(n - 1)/2$ edges (Diestel, 2005).

Let $G = (V, E)$ and $G = (V, E)$ be two graphs. Graph G is a *subgraph* of graph G if $V \subseteq V$ and $E \subseteq E \cap (V' \times V')$; accordingly, graph G is a *supergraph* of G'. We will show the subgraph relationship by $G \subseteq G$. By this definition, each graph is also a subgraph of itself.

We call G' and G *isomorphic* shown by $G' \sim G$, if there exists a bijection (one-to-one and onto) $f: V' \rightarrow V$ with $\langle u, v\rangle \in E' \Leftrightarrow \langle f(u), f(v)\rangle \in E$ for all $u, v \in V'$. Such a mapping f is called a *graph isomorphism*. Note that an isomorphic relation between two graphs G and G' implies $|V| = |V'|$. An *automorphism* of a graph is a graph isomorphism relationship from the graph to itself.

If there exists an injective (one-to-one only) h: $V' \rightarrow V$ and both $V' \subseteq V$ and $\langle u, v\rangle \in E \Leftrightarrow \langle h(u), h(v)\rangle \in E$ are held, this mapping represents an *appearance* (or *occurrence*) of G' in G. Such a mapping h is called *subgraph isomorphism* from G' into G. In other words, mapping h represents an isomorphic relation between G' and a subgraph of G.

The number of appearances of graph G' in G is called *frequency* of G' in G and is the total number of distinct subgraph isomorphism mappings from G' to G. Note that there are three different concepts F_1, F_2 and F_3 of frequency (Schreiber and Schwöbbermeyer, 2005b). The first frequency concept F_1 relates to all mappings of a subgraph in the original network; this definition is similar to the one we have discussed and here we only refer to this frequency concept. The second concept F_2 consists of all mappings which are edge disjoint (two sets A and B are called disjoint whenever $A \cap B = \emptyset$). Finally, the frequency concept F_3 entails all mappings that are edge and node disjoint. Therefore, as Fig. 1 shows two frequency concepts F_2 and F_3 restrict the usage of elements of the graph. Here, a graph is called *frequent* in G, when its frequency (frequency concept F_1) is higher than a predefined threshold value Δ.

A randomized version R of graph G as *Null-model* is a random graph with some similar properties to G, e.g., same degree distribution of the nodes in G (Ciriello and Guerra, 2008). There is an ensemble $\Omega(G)$ of random graphs corresponding to G. We should choose N (a predefined value) random graphs uniformly from $\Omega(G)$ and calculate the frequency for a particular frequent subgraph G in G. If the frequency of G in G is higher than its average frequency in the N random graphs R_i (V_i, E_i), where $1 \leq i \leq N$, then we treat G as a *network motif* for G.

The algorithm Our algorithm uses a pattern growth approach (Schreiber and Schwöbbermeyer, 2005b). At each step, the algorithm takes one graph as a query graph and tries to find the frequency (number of appearances) of the query graph in the given network. To calculate the frequency of a particular query graph, we should take into account the structure of the query graph. If the current query graph is a supergraph of a previous query graph, we can exploit information about the mappings of the previous query graph. As a result, as long as existing information from formerly found mappings is maintained we can easily calculate the frequency of each graph (with the exception of trees as we will show), and hence reduce computation time. Therefore, we should incorporate an appropriate hierarchical organization into the algorithm for each graph size. We use a concept named *expansion trees* T_k for each $9 \geq k \geq 3$ that prepares, to prepare a hierarchical structure in the algorithm. Note that this concept is slightly similar to the *pattern tree* introduced by Schreiber and Schwöbbermeyer (2005b) in that they used this concept for all graph sizes. Contrary to the pattern tree which works well for frequency concepts F_2 and F_3, the expansion tree is applicable to the frequency concept F_1.

T_k is like an atlas to steer the running process of the algorithm. The expansion tree plays an important role in our algorithm by providing us with query graphs systematically from minimally k-sized connected query graphs (trees) to complete query graph.

The expansion tree We use information about previous (same size) query graphs which are subgraphs of current query graphs. We start the algorithm from minimally connected query graphs, because we can consider other size k graphs by expanding these minimally connected graphs through adding edges and exploit hitherto found mappings of these graphs for computing the existence of expanded graphs without using subgraph isomorphism. The simplest connected graphs with the minimum number of edges are namely trees. Therefore, we first calculate the frequency of trees of a given size in a network and then expand these trees edge by edge until we get a complete graph such that there is no room for new edges. To organize this bottom-up idea, we intro-

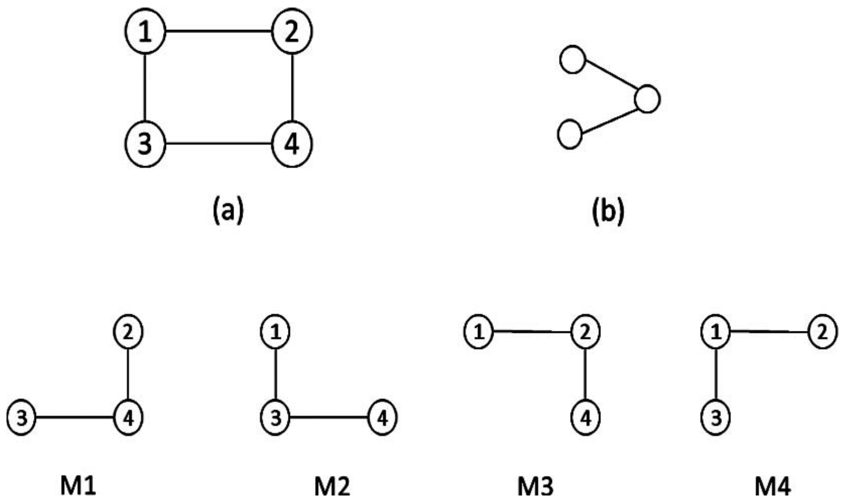


Fig. 1. Illustration of different frequency concepts by finding all mappings (M1–M4) of graph (b) into graph (a). For frequency concept F_1, the set {$M1$, $M2$, $M3$, $M4$} of matchings (mappings) is admissible, so $F_1 = 4$. For F_2 , one of two set {$M1$, $M4$} and {$M2$, $M3$} are acceptable and so $F_2 = 2$. Finally for frequency concept F_3, merely one of the matchings (M1–M4) is admissible; therefore, $F_3 = 1$.

duce the concept of expansion trees. Each node of the expansion tree is a graph that the algorithm uses as a query graph. The first level of expansion trees consist of trees (the root is in level 0, see Fig. 1). The query graphs become more complete by each step as the expansion tree is traversed in depth. Formally, the expansion tree T_k is a tree whose root is number k which indicates the graph size existing in the expansion tree, and has the following properties:

- Level of root assumed to be zero.
- Each node (except the root) consists of a graph of size k.
- At i^{th} level, each node includes a graph size k with $(k - 2 + i)$ edges.
- Number of nodes at the first level is equal to the number of non-isomorphic trees of size k.
- Each graph represented by a node is non-isomorphic to all other graphs in T_k.
- Each node (except the root) is a subgraph of its child.
- There is only one leaf at level $\frac{k^2-3k+4}{2}$ (the graph K_k a, complete graph with k (nodes).
- Longest path from root to leaf includes $\frac{k^2-3k+4}{2}$ edges.

Each node in the expansion tree stores an adjacency matrix that corresponds to a graph. Because we have here only considered simple undirected graphs, the corresponding adjacency matrices are *symmetric*, so we can only take into account entries either above or below the main diagonal of the adjacency matrix. Here, we adapt a slightly different definition of the adjacency code in the manner that is defined in Chen et al. (2006). The adjacency code of a symmetric matrix A, $CAD(A)$ is a sequence formed from entries below the main diagonal of A by placing them in the following order:

$$\alpha_{2.1}\,\alpha_{3.1}\,\alpha_{3.2}\,\alpha_{4.1}\,\alpha_{4.2}\,\alpha_{4.3}\,\alpha_{5.1}\,\cdots\,\alpha_{n.}(n-1)$$

In order to generate the expansion tree of a set of k-size non-isomorphic trees we follow a simple procedure. First, locate these distinct trees at the first level of the expansion tree. After that, expand the trees (or later the graphs) in subsequent levels by replacing a 0 entry by a 1 entry in the adjacency code. One can replace any 0 in the sequence of adjacency code by a 1 in each stage, which will yield to expanded graphs in which some are isomorphic. Note that the expanded graphs differ in merely one edge. If the expanded graphs are different then they represent different child nodes and if they are isomorphic, they represent the same child node. The expansion will finish when it obtains a complete graph in which the adjacency code is a sequence of 1s. At this stage, the constructed expansion tree from data structures point of view is in fact a directed acyclic graph, or a DAG. Afterwards, by running Depth-First Search algorithm on this graph, we obtain a tree in $\theta(V + E)$ which generally is called Topological Sort of the DAG or in MODA's terminology the Expansion Tree (Cormen et al., 2004).

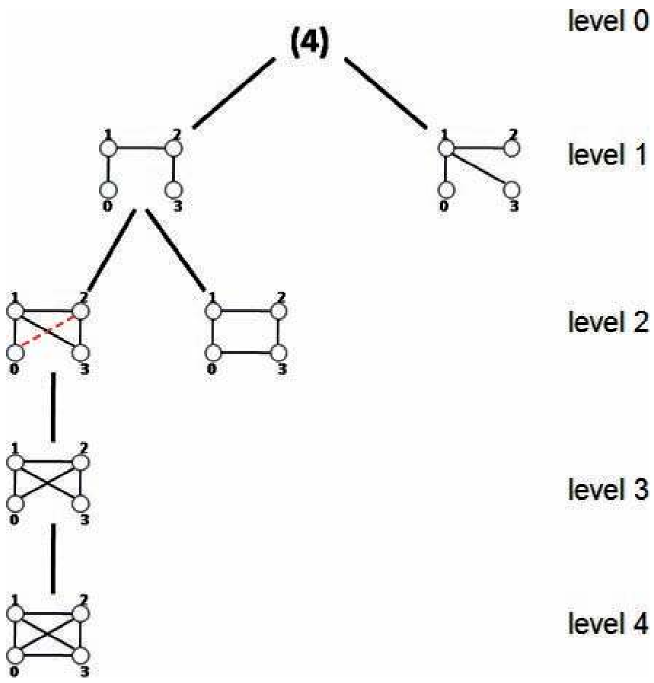


Fig. 2. The expansion tree T_4 for 4-node graphs. At the first level, there are non-isomorphic k-size trees and at each level an edge is added to the parent graph to form a child graph. All graphs in each level are non-isomorphic to prevent redundancy. In the second level, there is a graph with two alternative edges that is shown by a dashed red edge. In fact, this node represents two expanded graphs, let G be the graph with edge $\langle 0, 2\rangle$ and G' the graph with $\langle 1, 3\rangle$, such that the relation $G{\sim}G'$ is held between them and they differ only by a single edge. The depth of T_k is determined by a node that holds a complete graph of k nodes.

As a final point, the expansion trees are constructed for every graph size just once and can be used in each run; subsequently, they can be stored and retrieved whenever the algorithm needs. In other words, expansion tree is a *static* data structure and the algorithm does not have to generate during every run.

Calculating subgraph frequency To calculate the frequencies of k-size subgraphs, the algorithm employs T_k to direct the running progress in a bottom-up fashion, and as we will show in this section our algorithm calculates the frequency of each graph by two different methods, mapping and enumerating. In this manner, T_k provides query graphs hierarchically. Based on the level of T_k that the query graph is chosen from, the algorithm will decide how to determine the frequency of the query graphs. The algorithm traverses T_k in breadth-first manner; however, other methods (i.e. depth-first) may also be used. At first, the algorithm fetches the query graphs at the first level of T_k, (which are trees), then finds all mappings from these trees to the network using the mapping module (Mapping Module Section) and holds these calculated mappings in memory space for future use. Therefore, the frequencies of these trees in the network are calculated exactly; as we will show in the Sampling Throughout the Network section. After that, the query graphs at the second level of T_k are fetched and the appearance number of each of these query graphs, with respect to the stored mappings of their parent nodes in T_k, is computed by means of an enumerating module (Enumeration Module Section). Performing this part of the enumeration is much easier than enumerating the appearances of trees. Fortunately, the number of non-isomorphic k-size trees occupies a small portion of non-isomorphic k-size graphs when k is more than five (Fig. 3).

Pseudo-code of the algorithm for calculating frequency of k-size subgraphs is as follows

Algorithm 1 Find Subgraph Frequency (G, k, Δ)

Input: G: Input graph, k: subgraph size, Δ: threshold value

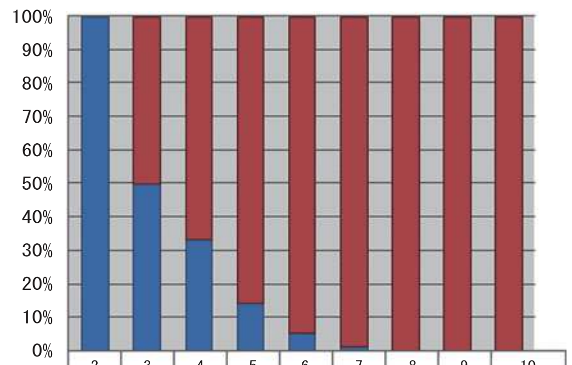
Output: Frequent Subgraph List: List of all frequent k-

Note: F_G: set of mappings from G in the input graph G

```
1: fetch $T_k$
2: do
3:     $G' = Get\text{-}Next\text{-}BFS(T_k)$   // $G'$ is a query graph
4:     if $|E(G')| = (k-1)$
5:             call Mapping-Module ($G'$, $G$)
6:         else
7:             call Enumerating-Module ($G'$, $G$, $T_k$)
8:     end if
9:     save $F_2$
10:    if $|F_G| > \Delta$ then
11:            add $G'$ into Frequent Subgraph List
12:    end if
13: Until ($|E(G')| = (k-1)/2)$)
14: return Frequent Subgraph List
```

In Algorithm 1, the function *Get-Next-BFS*(T_k) traverses T_k in a breadth-first manner and successively returns nodes. The two functions *Mapping-Module* and *Enumerating-Module* are the functions that actually enumerate the number of appearances of a query graph in the network. We will discuss these functions in the following sections. By definition, each subgraph that appears more often than a predefined threshold value is treated as a frequent subgraph; lines 10–12 perform this task. Line 13 checks the termination condition of the loop by considering the structure of query graph G. If G is a complete graph of order k then the loop terminates.

The mapping module For a query graph, there is a set of mappings denoted by $F_{G'}$ which contains different mappings from G' to the network or nothing if there is no mapping from G' to the network. Each mapping for



	2	3	4	5	6	7	8	9	10
🟥 graphs	1	2	6	21	112	853	11117	261080	11716571
🟦 trees	1	1	2	3	6	11	23	47	106

Fig. 3. The diagram shows the percentage of trees (blue) in k-size connected graphs for $2 \le k \le 10$, with *non-tree* graphs shown in red color. The table below the diagram shows how many of the total k-size graphs are trees in different sizes of k (shown by blue). By increasing k, the number of trees increases more slowly in comparison with exponential growth in the number of non-tree graphs. (Junker and Schreiber, 2008).

G' to the network with node set V, is a function $f_{G'}^{i} : V' \to V$, where $1 \le i \le |F_{G'}|$. The frequency of query graph G' is determined by $|F_{G'}|$. From a computational complexity point of view, finding a mapping from G' to graph G is called the subgraph isomorphism problem and is *NP*-complete (Garey and Johnson, 1979). For that reason, finding all possible mappings from G' to G is a counting problem and is in #*P*-complete (Papadimitriou, 1994). Hence, there is no known algorithm to solve this problem in polynomial time. We use a well-known algorithm that tries to find mappings in a branch and bound manner. To speed up the algorithm, we exploited *symmetry-breaking conditions* as a heuristic. Using symmetry-breaking conditions, the overcounting problem is solved because it causes the enumeration of only one of the several possible mappings that may exist from the query to a subset of nodes in the network and leaves out all other extras. This module is the one that is used in the *Grochow-Kellis* algorithm (Grochow and Kellis, 2007) and exhaustively counts all mappings for a given query graph. We incorporated a graph invariant that is used in *Grochow-Kellis,* which can be evaluated for each node efficiently and is consequently beneficial in practice. Some other invariants (e.g., neighbors' degrees for each node) were tested, and most of them resulted in overhead and imposed extra computational costs.

The pseudo-code of the mapping-module is as follows,

Algorithm 2 Mapping-Module (G', G)

Input: G: Input graph, G': Query graph

Output: F_G: List of all mappings from G' to G

```
1: $F_G = \emptyset$
```

```
2: for $i = 1$ to predefined-value do
3:     select node $v$ from $G$ with a probability proportional
4:     foreach node $u$ of $G'$ do
5:         if $k_v \geq k_u$ then
6:             set $f(u) = v$
7:             find all of isomorphism mappings
8:             add mappings into $F_G$
9:         end if
10:    end for
11:    remove $v$ from $G$
12: end for
13: return and save $F_G$
```

In Algorithm 2, the module returns a list of all mappings from G to G. In line 2, the *predefined-value* indicates the number of samples of the network. For an exact enumeration of all appearances of a query graph one can simply set *predefined-value* equal to the network size or $|V(G)|$. We set *predefined-value* to $|V(G)|/3$ for all of tests in the Runtime Analysis section. In line 3, the algorithm chooses a node with probability equal to Eq. 1 (Sampling Throughout the Network Section). In lines 4–10, the algorithm tries to compute all possible mappings from G to G that contain the chosen node (in line 3) in their range. In line 5, the algorithm uses a simple form of invariant as a heuristic to stop the search for a mapping when no mapping exixts. One can include other efficient invariants into this step to enhance the performance of the algorithm. In Grochow and Kellis (2007) more information is provided about line 7 that leads to the determination of all mappings from the query graph to the network by means of a branch and bound algorithm. Removing the chosen node v in line 11 helps to prevent finding redundant mappings that have been found in previous steps.

We use this module for query subgraphs at the first level (trees) of T_k. For this, we hold any mapping set F_G corresponding to each G at the first level of T_k in predetermined memory space. Afterward, for any other query graphs in T_k we will not call this module and therefore will not calculate the frequency of these graphs. As a result of calling the mapping module solely for *trees* some improvements can be made in the algorithm.

The enumeration module In Fig. 2 it is clear that after the first level of T_k each node only adds a single edge to its parent to create a new graph that is a supergraph of its parent and non-isomorphic to its counterparts (same level nodes in T_k). Up to this point, we have computed all mappings for trees in the first level of T_k and kept this information. Now, we exploit this information with respect to the following fact:

If there was a mapping f_G from $G = (V, E)$ to the network then f_G also maps $G' = (V',E')$ to the network, whenever $G' \subseteq G \wedge |V| = |V'|$ holds

Note that the reverse of the above fact does not necessarily hold and the condition $|V| = |V'|$ is essential. Therefore, it would be helpful to give queries to the network hierarchically, and T_k serves this idea for us by traversing the expansion tree in breadth-first order. Suppose we want to calculate the frequency of a query graph G that is located at the second level of T_k, we can extract the mapping set F_G corresponding to its parent node, then enumerate all mappings in F_G that can support G and import them into F_G. What criteria must be met by the mapping f_G^i to support G? Without loss of generality, let $\langle u, v\rangle$ be the new edge in G. If there exists an edge $\langle f_G^i(u), f_G^i(v)\rangle$ in the network, which does not lead to the violation of the symmetry-breaking conditions of G as well, then f_G^i can support G, consequently $f_G^i \in F_{G'}$. As a result, we have in this case exchanged the subgraph isomorphism problem with a simple criterion, which can be examined in $O(1)$.

It is possible to have multiple candidate edges for making a particular graph from another graph. For instance, in Fig. 2 we have two edges (dashed red and black ones), so we have two choices for each mapping. We consider each candidate edge separately. In this case, when a single mapping can fulfill n conditions as well as the symmetry-breaking conditions for each new edge, the algorithm enumerates that mapping n times and adds n number of this mapping to $F_{G'}$. Yet this approach is better than an exhaustive search. For instance, when we would like to find the frequencies of graphs with G nodes, we have to find six non-isomorphic trees exhaustively and 112 dissimilar graphs via this new approach (Fig. 3), which brings significant savings in running time in comparison with a straight forward solution that enumerates all 6-node subgraphs.

Algorithm 3 Enumerating-Module (G', G, T_k)

Input: G: Input graph, G': Query graph, T_k: expansion tree

Output: F_G: List of all mappings from G' to G

```
1: $F_G = \emptyset$
2: $H$ = Get-Parent ($G'$, $T_k$)
3: load $F_H$ //    $F_H$ was stored in the memory
4: let $\langle u, v\rangle = E(G')-E(H)$
5: foreach $f \in F_H$ do
6:     if ($\langle f(u), f(v)\rangle \in G$) AND $\langle f(u), f(v)\rangle$
       violates the corresponding
       conditions then
7:         add $f$ into $F_{G'}$
8:     end if
```

```
9: end for
10: return $F_G$
```

As in Algorithm 2, Algorithm 3 returns a list of all mappings from G to G In line 2, the algorithm simply finds the parent node of G and loads its mapping set that was found in previous steps. Line 4 determines the new edge that has been added by G to its parent. Lines 5–9 decide which mapping can support defined criteria.

In order to count only induced subgraphs, for each mapping the algorithm tries to expand the mapping as deep as possible in the expansion tree. Then, for each specific mapping, the algorithm simply counts the appearance of the latest node in the expansion tree that the mappings reached and discards all other nodes before that node of the tree. By this approach, the algorithm does not consider subgraphs of an induced subgraph in the network and only counts induced subgraphs.

Sampling throughout the network Despite the fact that the mapping module is used for a small portion (trees) of k-size non-isomorphic graphs (Fig. 3), it still takes a lot of time. To speed up this part of the algorithm, we ought to incorporate an appropriate *sampling method* within the mapping module (Mapping Module Section).

By sampling throughout the network, we do not have to find all of the mappings in which a whole set of nodes participate. Instead, we select only a subset of nodes in the network in the mapping module. However, the selection method must ensure reliable results. This means the number of mappings that we find for a given query graph must be similar in different runs of the algorithm. Therefore, we propose a probability distribution on V that leads to reliable results for every run of the algorithm. Empirically, uniform distribution on V does not fulfill the condition about reliability and exhibits large fluctuations (blue line in Fig. 4).

However, Vázquez et al. (2004) have shown the power-law degree distribution influence on scattering subgraphs around the network. In other words, the subgraph aggregation around the high degree nodes (hubs) is higher than around low degree nodes. We investigated the distribution of subgraphs around nodes for some real-world networks (Supplementary I, http://LBB.ut.ac.ir/Download/LBBsoft/MODA/) as evidence for this conjecture. Empirical results have shown the number of subgraphs in which a node participates is proportional to its degree. Therefore, we utilized the node's degree as selection criteria for sampling throughout the network. Formally, the probability of selecting an arbitrary node with degree k_i in the network is given by

$$\Pr(v_i)\frac{k_i}{\sum_j k_j} \qquad (1)$$

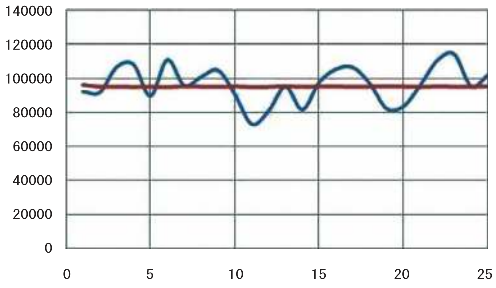


Fig. 4. Fluctuations of results in two approaches of sampling for a particular query graph. The x-axis indicates different runs and the y-axis frequency for each run. The blue line shows the sampling method based on uniform sampling. The red line stands for the sampling approach that relies on degree distribution of nodes in the network; in other words, the probability of sampling each node is determined by Eq. 1 which is proportional to the degree of each node. This approach shows little fluctuation in results for different runs.

where j is an index over all nodes of the network. As a result, this sampling leads to low fluctuation in results for different runs, and hence reliable results (Fig. 4). Equation 1 is identical in probability distribution as the one in the *preferential attachment* mechanism that was employed in the Barabási-Albert model (Barabási and Albert, 1999); this mechanism generates scale-free networks.

As a final point, the sampling method that we have discussed above would increase the efficiency of the algorithm; however, to the detriment of accuracy.

RESULTS

Runtime analysis All runtime analyses were performed on an IBM R50e laptop with Intel Pentium 1.8 GHz and 1 GB RAM, using the Windows XP Home Edition operating system. Source codes were compiled with .NET 2.0. The loading time of the network was not included. The network instance for testing our algorithm was the *E. coli* transcription network (Shen-Orr et al., 2002) that is taken from (http://www.weizmann. ac.il/mcb/UriAlon/coliData.html). We considered only the frequency of each subgraph without considering random networks as Null-models since we aimed to assess the computational time of the algorithm in the enumeration of subgraph appearances and not to determine occurrences of motifs.

Comparing the runtime of the pure method versus sampling method The previous section has shown that sampling increases the efficiency of the algorithm since it prevents a complete search of the search space of the problem instance. Both sampling and exact approaches were implemented and their comparison is depicted in Figs. 4 and 5. Although the number of k-size trees is negligible in comparison with k-size non-tree graphs (Fig. 3), it takes an enormous amount of time to compute the frequency of all k-size trees. For graphs with more than nine nodes, this becomes computationally difficult with state-of-the-art computer technology. We solved this problem by a sampling approach; nonetheless, for larger graph sizes even this approach will be infeasible since the number of potential query graphs which must be considered by the algorithm will increase exponentially (Inokuchi et al., 2000; Kuramochi and Karypis, 2004). These results demonstrate that the sampling approach is unable to find motifs larger than ten nodes. It seems that some new approaches must be considered to find larger sized motifs, parallel computing may provide a future solution to overcome this problem.

The effect of the enumerating module We captured the runtime of the algorithm through different steps for computing the frequencies for six unique undirected graphs with four nodes (Fig. 6). After computing frequencies for the first two graphs that are trees, the runtime of MODA immediately falls for the remaining graphs.

In Fig. 6, the runtime of *Grochow-Kellis* decreases gradually since the number of appearances of dense graphs decreases. For instance, in the *E.coli* network there is no complete graph of four nodes. For the MODA algorithm without sampling the runtime rapidly falls close to zero for different query graphs (not trees) because of the sophisticated method for providing query graphs. Additionally, in Fig. 6 the runtime for the first two query trees noticeably decreases when the algorithm utilizes the sampling method (see “sampling throughout the network” section).

Runtime comparison between MODA and other algorithms We compared MODA with the *Grochow-Kellis, mfinder, FANMOD, FPF* algorithms (Fig. 7). In this study, the frequencies of non-induced as well as induced subgraphs were determined by MODA. We would have liked to compare MODA with *NeMoFinder*, but could not find an implementation of the *NeMoFinder* algorithm. Note that all of the results consider only enu-

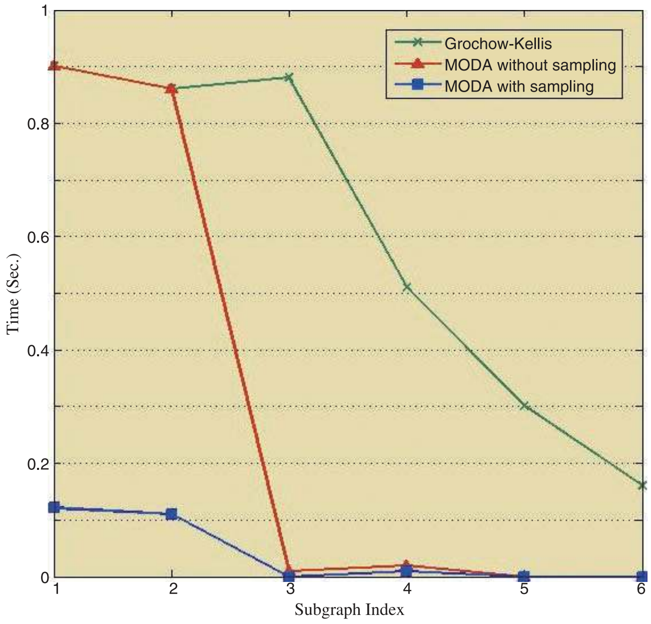


Fig. 6. Comparison of running times for different methods. × indicates the runtime of the *Grochow-Kellis* algorithm. ▲ indicates the runtime of our algorithm (MODA) without sampling. indicates the runtime of MODA with sampling. ■ The numerical labels in the x-axis of the query graphs are taken from the breadth-first traverse of the expansion tree in Fig. 2.

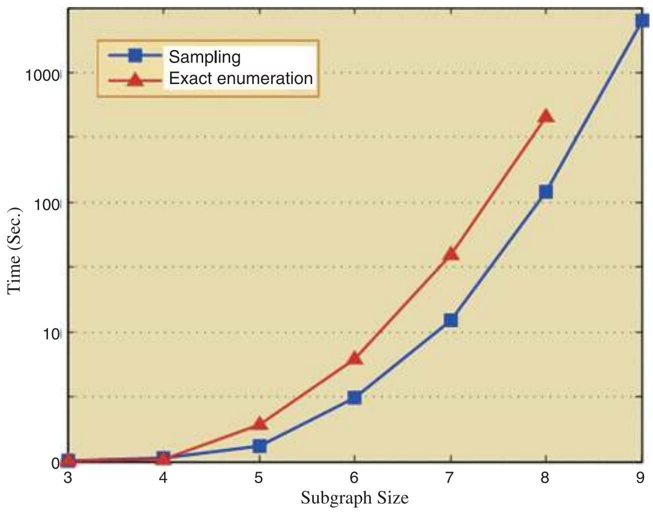


Fig. 5. Runtime comparison between two versions of MODA: the sampling and the exact approach. 0.25 portions of nodes were taken as samples in the sampling method.

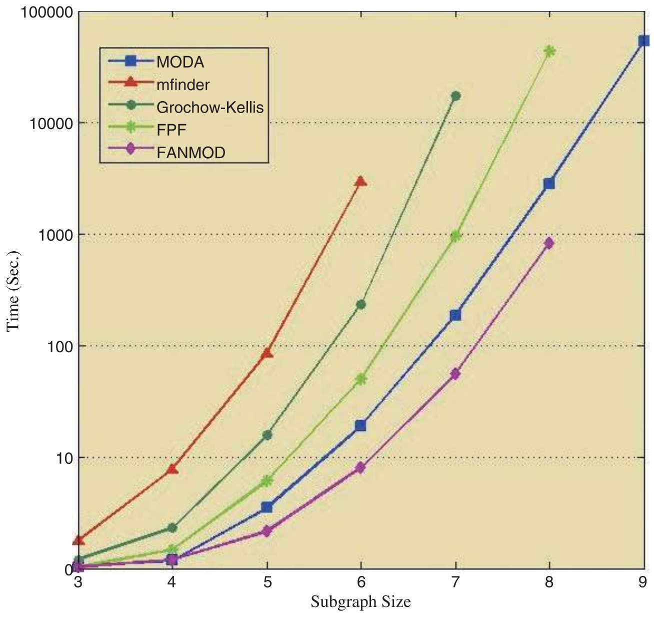


Fig. 7. Runtimes of *Grochow-Kellis*, *mfinder*, *FANMOD, FPF* and *MODA* for subgraphs from three nodes up to nine nodes.

meration in the network, and computational time for randomized networks is not included. The time for computing frequencies in random networks is same as for the input network. From Computational complexity point of view, considering random networks multiply computational time by a constant coefficient.

DISCUSSION

There is a great deal of research activity in studying different aspects of complex networks. Complex networks display diverse properties at different levels. High clustering coefficients or degree distributions are some of these features at the macroscopic level. On the other hand, important microscopic (local) properties exist as well as global ones. One of the important local properties of complex networks is frequent subgraphs that are statistically over-represented in a network. Such subgraphs, which are more frequent in a real-world network than in its randomized counterparts are called network motifs. Despite the fact that the definition of network motifs does not give priority to induced subgraphs over non-induced subgraphs, the purpose of network motifs is in practice often induced subgraphs.

Extracting motifs from a large real-world network demands high computational effort to cope with *NP*-complete problems. The problem of finding frequent patterns in databases has been studied in the field of data mining. Hence, implementing some ideas from data mining in network motif discovery may be fruitful. In this paper, we have suggested a method that relies on the pattern growth approach. This approach leads to systematic querying from the network and exploiting previous results of preceding queries. These results helped us to determine the frequency of each *non-tree* graph without performing the *subgraph isomorphism* that is *NP*-complete.

In addition, many of the real-world networks are very large graphs and finding all of the mappings (matches) for a relatively small query graph in these networks is extremely difficult. For that reason, we should obtain samples from the network and not exhaustively count the appearances of each query graph. Nevertheless, the outcomes of the sampling should not lead to strong fluctuations in different runs. We used the degree of each node as a yardstick to obtain samples from the network, resulting in similar results in any execution. Using these methods, we were able to find relatively large motifs in efficient time. The MODA source code is freely available at: http://LBB.ut.ac.ir/Download/LBBsoft/MODA/

We thank Joshua A. Grochow for critical reading of the manuscript and helpful suggestions. We also thank Farid Alaghmand for his support during this work. Part of this work was supported by Iran National Science Foundation (http://www.insf.org/).

REFERENCES

Agrawal, R., and Srikant, R. (1994) Fast Algorithms for Mining Association Rules in Large Databases. In: Proceedings of 20th International Conference on Very Large Data Bases (VLDB'94), pp. 487–499. Morgan Kaufmann, Santiago de Chile, Chile.

Alon, U. (2007) Network motifs: theory and experimental approaches. Nature Review Genetics **8**, 450–461.

Amaral, L. A., Scala, A., Barthelemy, M., and Stanley, H. E. (2000) Classes of small-world networks. Proc. Natl. Acad. Sci. USA **97**, 11149–11152.

Barabási, A. L., and Albert, R. (1999) Emergence of scaling in random networks. Science **286**, 509–512.

Baskerville, K., and Paczuski, M. (2006) Subgraph Ensembles and Motif Discovery Using a New Heuristic for Graph Isomorphism. Phys. Rev. E. Stat. Nonlin. Soft Matter Phys. **74**, 051903.

Bornholdt, S. S., and Schuster, H. G. (2003) Handbook of Graphs and Networks: From the Genome to the Internet. Wiley-VCH, Weinheim.

Chen, J., Hsu, W., Lee, M.–L., and Ng, S. K. (2006) NeMoFinder: dissecting genome-wide protein-protein interactions with meso-scale network motifs. In: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'06), pp. 106–115. ACM SIGKDD, Philadelphia.

Ciriello, G., and Guerra, C. (2008) A review on models and algorithms for motif discovery in protein-protein interaction networks. Brief. in Func. Genomics and Proteomic **7**, 147–156.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2004) Introduction to Algorithms. MIT Press, Massachusetts.

Diestel, R. (2005) Graph theory. Springer-Verlag, Heidelberg.

Dunne, J. A., Williams, R. J., and Martinez, N. D. (2002) Food-web structure and network theory: The role of connectance and size. Proc. Natl. Acad. Sci. USA **99**, 12917-12922.

Erdös, P., and Rényi, A. (1960) The Evolution of Random Graphs. Magyar Tud. Akad. Mat. Kutató Int. Közl. **5**, 17–61.

Faloutsos, M., Faloutsos, P., and Faloutsos, C. (1999) On power-law relationships of the internet topology. In: Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM '99), pp. 251–262. ACM SIGCOMM , Massachusetts.

Garey, M. R., and Johnson, D. S. (1979) Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman, New York.

Grochow, J. A., and Kellis, M. (2007) Network Motif Discovery Using Subgraph Enumeration and Symmetry-Breaking. In: Proceedings of International Conference on Research in Computational Molecular Biology (RECOMB' 07), pp. 92–106. Springer-Verlag, Heidelberg.

Huan, J., Wang, W., and Prins, J. (2004) SPIN: mining maximal frequent subgraphs from graph databases. In: Proceedings of the tenth SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'04), pp. 581–586. ACM SIGKDD, Seattle.

Inokuchi, A., Washio, T., and Motoda, H. (2000) An Apriori-based algorithm for mining frequent substructures from graphs. In: Proceedings of the 4th European Conference on Principles of Data Mining and Knowledge Discovery (PKDD), pp. 13–23. Lyon, France.

Itzkovitz, S., Milo, R., Kashtan, N., Ziv, G., and Alon, U. (2003)

Subgraphs in random networks. Phys. Rev. E **68**, 026127.

Junker, B. H., and Schreiber, F. (2008) Analysis of biological networks. Wiley Series on Bioinformatics, Computational Techniques and Engineering. Wiley-VCH, Weinheim.

Kanehisa, M., Araki, M., Goto, S., Hattori, M., Hirakawa, M., Itoh, M., Katayama, T., Kawashima, S., Okuda, S., and Tokimatsu, T., et al. (2008) KEGG for linking genomes to life and the environment. Nucleic Acids Res. **36**, 480–484.

Kashtan, N., Itzkovitz, S., Milo, R., and Alon, U. (2004) Efficient sampling algorithm for estimating subgraph concentrations and detecting network motifs. Bioinformatics **20**, 1746–1758.

Kuramochi, M., and Karypis, G. (2004) An efficient algorithm for discovering frequent subgraphs. IEEE Transactions on Knowledge and Data Engineering (*TKDE*), 1038–1051.

Mangan, S., and Alon, U. (2003) Structure and function of the feed-forward loop network motif. Proc. Natl. Acad. Sci. USA **100**, 11980–11985.

Masoudi-Nejad, A., Goto, S., Jauregui, R., Ito, M., Kawashima, S., Moriya, Y., Endo, T., and Kanehisa, M. (2007) EGENES: transcriptome-based plant database of genes with metabolic pathway information and expressed sequence tag indices in KEGG. Plant Physiol. **44**, 857–866.

Middendorf, M., Ziv, E., and Wiggins, C. H. (2005) Inferring network mechanisms: the *Drosophila melanogaster* protein interaction network. Proc. Natl. Acad. Sci. USA **102**, 3192–3197.

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

Ouzounis, C. A., and Karp, D. P. (2000) Global properties of the metabolic map of *Escherichia coli*. Genome Res. **10**, 568–576.

Papadimitriou, C. H. (1994) Computational complexity. pp. 439–447. Addison-Wesley, Massachusetts.

Przytycka, T. M. (2006) An important connection between network motifs and parsimony models. Lecture Notes in Computational Biology, pp. 321–335. Springer-Verlag, Heidelberg.

Razaghi Moghadam Kashani, Z., Ahrabian, H., Elahi, E., Nowzari-Dalini, A., Saberi Ansari, E., Asadi, S., Mohammadi, S., Schreiber, F., and Masoudi-Nejad, A. (2009) Kavosh: a new algorithm for finding network motifs. BMC Bioinformatics **10**, 318.

Schreiber, F., and Schwöbbermeyer, H. (2005a) MAVisto: a tool for the exploration of network motifs. Bioinformatics **21**, 3572–3574.

Schreiber, F., and Schwöbbermeyer, H. (2005b) Frequency concepts and pattern detection for the analysis of motifs in networks. In: Transactions on Computational Systems Biology III, pp. 89–104. Springer-Verlag, Heidelberg.

Shen-Orr, S., Milo, R., Mangan, S., and Alon, U. (2002) Network motifs in the transcriptional regulation network of *Escherichia coli*. Nature Genetics **31**, 64–68.

Tsang, J., Zhu, J., and van Oudenaarden, A. (2007) MicroRNA-mediated feedback and feedforward loops are recurrent network motifs in mammals. Mol. Cell **26**, 753–767.

Vázquez, A. (2002) Degree correlations and clustering hierachy in networks: measures, origin and consequences. PhD Dissertation, La Scuola Internazionale Superiore di Studi Avanzati / International School for Advanced Studies, Trieste, Italy.

Vázquez, A., Dobrin, R., Sergi, D., Eckmann, J. P., Oltvai, Z. N., and Barabási, A. L. (2004) The topological relationship between the large-scale attributes and local interaction patterns of complex networks. Proc. Natl. Acad. Sci. USA **101**, 17940–17945.

Watts, D. J., and Strogatz, S. H. (1998) Collective dynamics of 'small world' networks. Nature **393**, 409–410.

Wernicke, S. (2006) Efficient detection of network motifs. IEEE/ACM Trans. Computational Biology and Bioinformatics **3**, 347–359.