Libertas Academica
FREEDOM TO RESEARCH

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

Evolutionary Bioinformatics

A New Method for Motif Mining in Biological Networks

Yuan Xu, Qiang Zhang and Changjun Zhou

Key Laboratory of Advanced Design and Intelligent Computing, Dalian University, Ministry of Education, Dalian, China.

ABSTRACT: Network motifs are overly represented as topological patterns that occur more often in a given network than in random networks, and take on some certain functions in practical biological applications. Existing methods of detecting network motifs have focused on computational efficiency. However, detecting network motifs also presents huge challenges in computational and spatial complexity. In this paper, we provide a new approach for mining network motifs. First, all sub-graphs can be enumerated by adding edges and nodes progressively, using the backtracking method based on the associated matrix. Then, the associated matrix is standardized and the isomorphism sub-graphs are marked uniquely in combination with symmetric ternary, which can simulate the elements (–1,0,1) in the associated matrix. Taking advantage of the combination of the associated matrix and the backtracking method, our method reduces the complexity of enumerating sub-graphs, providing a more efficient solution for motif mining. From the results obtained, our method has shown higher speed and more extensive applicability than other similar methods.

KEYWORDS: sub-graphs mark, associated matrix, backtracking, symmetric ternary

CITATION: Xu et al. A New Method for Motif Mining in Biological Networks. *Evolutionary Bioinformatics* 2014:10 155–163 doi: 10.4137/EBO.S15207.

RECEIVED: March 4, 2014. **RESUBMITTED:** July 18, 2014. **ACCEPTED FOR PUBLICATION:** July 24, 2014.

ACADEMIC EDITOR: Jike Cui, Associate Editor

TYPE: Methodology

FUNDING: This work is supported by the National Natural Science Foundation of China (No. 31370778) and by the Program for Changjiang Scholars and Innovative Research Team in University (No. IRT1109). The authors confirm that the funder had no influence over the study design, content of the article, or selection of this journal.

COMPETING INTERESTS: Authors disclose no potential conflicts of interest.

CORRESPONDENCE: zhangq@dlu.edu.cn

This paper was subject to independent, expert peer review by a minimum of two blind peer reviewers. All editorial decisions were made by the independent academic editor. All authors have provided signed confirmation of their compliance with ethical and legal obligations including (but not limited to) use of any copyrighted material, compliance with ICMJE authorship and competing interests disclosure guidelines and, where applicable, compliance with legal and ethical guidelines on human and animal research participants.

Introduction

After the Human Genome Project (HGP), we have entered a "post-gene era", in which researchers have discovered that the human genome is not a combination of isolated genes and a large amount of useless "DNA segments," but a complex network system. Network motifs are overly represented as the smallest unit in a network.[1–3] Motif analysis is increasingly recognized as a powerful approach to research the function structure of a network, the structure of the organization principle, and species evolution.

Network motifs were first defined systematically in *Escherichia coli*. Following this work, many methods of network motif mining have sprung up. Kashtan et al proposed edge sampling algorithm (ESA),[4] which unfortunately leads to sampling deviation and causes error. Wernicke described an algorithm to enumerate sub-graphs, named Enumerating Subgraph (ESU), which solves the defects of ESA and improves the operation efficiency.[5] Meanwhile, a model based on generating random networks has also been found to be a key to identifying motifs. Milo et al proposed a go-with-the-winner algorithm to generate random networks.[6] As sub-graph isomorphism is an NP-complete problem,[7] a method of overcoming this by reducing the search size was introduced by Ding and Huang.[8] An algorithm aiming to reduce the complexity of matching two graphs was proposed by Knossow et al,[9] and then another algorithm that optimized one-to-many matching problems was described by Ogras and Marculescu.[10] In addition, based on the aforementioned algorithms, *MFINDER*, *PAJEK*, *MAVOSTO*, and *FANMOD* are notable existing tools for the motif-mining problem.[4,11–13] *MFINDER* was the first motif-mining tool, and it implements two kinds of motif-finding algorithms. *PAJEK* offers limited functionality, finding only specific motifs such as triads and particular tetrads in a network.

FANMOD can handle sub-graphs consisting of up to eight nodes.[13] However, these methods designed for both directed and undirected graphs were time consuming. In this paper, we present a method for reducing the searching time and saving storage space, while still storing all sub-graphs.

Our method counts all sub-graphs that meet the requirements in a given graph. Backtracking method is utilized to enumerate all sub-graphs in the network. Then, the associated matrixes of sub-graphs are normalized and the isomorphism sub-graphs are marked uniquely. We evaluate our method on the metabolic pathway of the bacteria *E. coli*, the transcription network of yeast, the sea urchin network, and an electronic network. Through these evaluations, we verify the correctness and extensive applicability of our method. In addition, we provide a new scheme of the associated matrix that is used to identify the motif in the digraph domain.

Background information

Definition of *network*. A network consists of vertices and edges. A directed graph (or network) is usually denoted by $G = (V,E)$, where V represents a finite set of nodes and E a finite set of edges such that $E \subseteq (V \times V)$. An edge $e = (u,v) \subseteq E$ goes from the node u (the source) to another node v (the target).

Definition of *induced sub-graph*. Two graphs: $G = (V,E)$, $G_s=(V_s,E_s)$, if and only if $V_s \subseteq V$ and $E_s \subseteq E$, then graph G_s is a sub-graph of G.

Definition of *degree order*. To order degrees of graph $G = (V,E)$, $V = \{v_1,v_2,\ldots,v_n\}$, sort all vertices V, by degree from big to small. That means graph's degree is ordered. D_a is the alignment of the graph.

Definition of *sub-graph isomorphic*. Two sub-graphs $G=(V_1,E_1)$ and $G'=(V_2,E_2)$ are isomorphic if there is a one-to-one correspondence between their vertices $f:v_1 \rightarrow v_2$, and there is an edge $g:e_1 \rightarrow e_2$ directed from one node to another node of one sub-graph $e_1 \in E_1, \Psi_1(e) = <u,v>$ if and only if there is an edge with the same direction between the corresponding vertices in the other sub-graph $\Psi_2(g(e)) = <f(u),f(v)>$.[14] Figure 1 shows the isomorphic visually.

The key of motif mining. There are five subtasks in our method: *form of storage* – a way to store digraph and undirected graph; *backtracking* – enumerating all sub-graphs of a given size that occur in the input graph; *random graph generation* – generating random graphs that meet the requirement of the input network; *sub-graph isomorphic* – standardizing the associated matrix, which marks the graph uniquely; and *motif identification* – distinguishing motifs among all founded sub-graphs on the basis of statistical parameters (Fig. 2).

The steps of mining motifs are as follows:

Use the ReadGraph function to store real graphs and random graphs.

Enumerate all of the sub-graphs with the backtracking algorithm.

Standardize all types of sub-graphs to easily identify the sub-graphs of the same type. We calculate the Code for a sub-graph based on the degree of sub-graphs and symmetric ternary; one Code will represent one type of sub-graph. We obtain the number of one type of sub-graphs based on the value of Code.

These steps are described in detail below.

Associated matrix. Storage of graphs is the first step in the process to solve the problem of motif mining. An associated matrix is easy to compress and makes it easy to know the size of the sub-graph; hence, we use it to store the real graphs and random graphs.

Definition of *associated matrix*. An associated matrix consists of a graph $G = <V,E>$, a set of nodes $V = \{v_1,v_2,\ldots,v_i\}$, and a set of edges $E = \{e_1,e_2,\ldots,e_j\}$, $1 \le i \le$ the number of graph's nodes and $1 \le j \le$ the number of graph's edges. $M(G)$ shows the connection between the nodes and edges of G. It is usually not a square matrix (Fig. 3).

As we know a row represents a node, a column represents an edge, and the algebraic sum of all elements is zero in $M(G)$. In the matrix $M(G)$, the sum of out-degree and in-degree is both equal to the number of edges, that is

$$\sum_{i=1}^{n}\sum_{j=1}^{m}(m_{ij}=1) = m = -\sum_{i=1}^{n}\sum_{j=1}^{m}(m_{ij}=-1).$$

The standardization of associated matrix. Because the sort of a graph's nodes is not sure, the associated matrix used

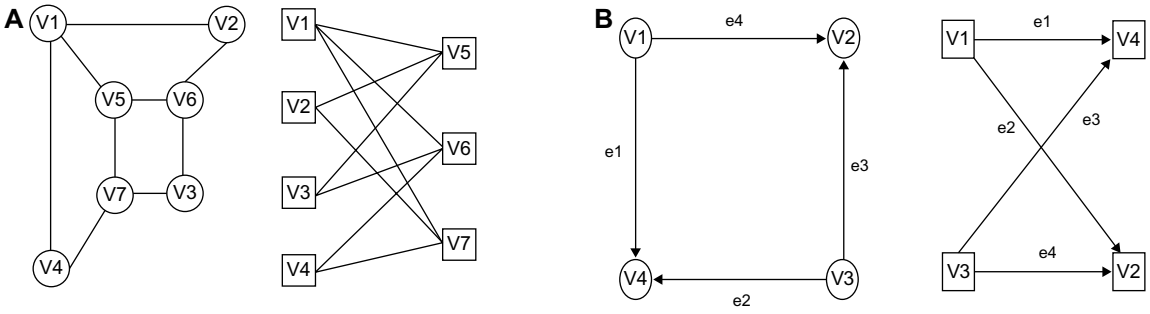


Figure 1. Examples of isomorphic graphs. (**A**) The isomorphism of an undirected graph. (**B**) The isomorphism of a directed graph.

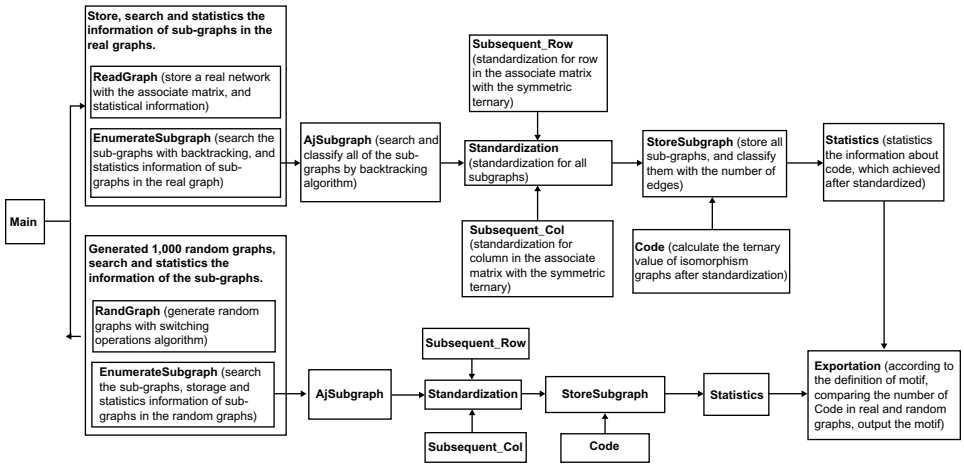
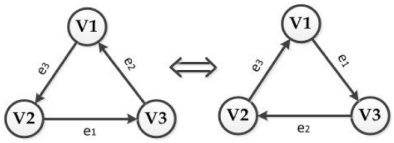


Figure 2. Flow chart of mining motifs.

to store a graph is not unique. In order to mark the graph uniquely, we need to standardize the associated matrix. This is useful for sub-graph isomorphism. Specifically, it can reduce the complexity of sub-graph isomorphism and improve the efficiency of searching. These will be discussed in detail in the next section.

The detailed processes of the standardization of an associated matrix are as follows:

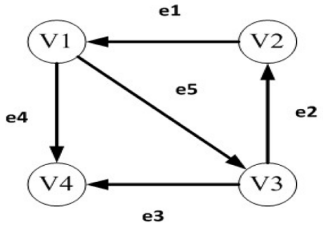
1. Out-degree order: perform a sort to all vertices with the dictionary sequence, which is in descending order.
2. In-degree order: make a dictionary sequence of the vertices having the same out-degree order.
3. If and only if there are some nodes with the same out-degree order and in-degree order, the sequence of these nodes should be in accordance with the minimum number of edges connecting them and the numbers shall ascend in the sequence.
4. Calculate the columns by symmetric ternary number, and then with the dictionary sequence. (All columns of ternary numbers are not the same.)
5. After sequencing these nodes, we shall transform the rows into a one-dimensional array and get a decimal number through transformation of the ternary number.
6. We may repeat the aforementioned steps until the matrices get the same *Code* number before and after transformation.



$$\begin{array}{c|ccc} & e1 & e2 & e3 \\ v1 & 0 & -1 & 1 \\ v2 & 1 & 0 & -1 \\ v3 & -1 & 1 & 0 \end{array} \Leftrightarrow \begin{array}{c|ccc} & e2 & e3 & e1 \\ v3 & 1 & 0 & -1 \\ v2 & -1 & 1 & 0 \\ v1 & 0 & -1 & 1 \end{array}$$

$$\Leftrightarrow (10-1-1100-11)_{\text{Symmetric Ternary}} = 5668$$

$$m_{ij} = \begin{cases} 1 & \text{if } v_i \text{ is the starting node of } e_j, \\ -1 & \text{if } v_i \text{ is the end node of } e_j, \\ 0 & \text{if } v_i \text{ is not associated with } e_j, \end{cases} \quad (i, j \text{ is the row and column of matrix})$$



$$M(G) = \begin{array}{c|ccccc} & e_1 & e_2 & e_3 & e_4 & e_5 \\ v_1 & -1 & 0 & 0 & 1 & 1 \\ v_2 & 1 & -1 & 0 & 0 & 0 \\ v_3 & 0 & 1 & 1 & 0 & -1 \\ v_4 & 0 & 0 & -1 & -1 & 0 \end{array}$$

Figure 3. An example of an associated matrix. It shows the corresponding relationship between a graph and its form for storage.

Finally, we get a Code value to mark the class of isomorphism. 5668 is a Code signing an isomorphism sub-graph.

Random network generation. For the identification of a motif, the proper determination of a sub-graph's significance needs comparison with an ensemble of random graphs. Generation of this ensemble for a given graph model is a necessary step in our method. We generate random graphs with the same feature of the input network (enumeration and classification are also performed on random graphs).

Our approach is based on switching operations,[15] applied on the edges of the input network repeatedly, until the network is well randomized. This switching operation is applied to randomly chosen vertices of the network, as shown in Figure 4. Any two edges ($A \rightarrow B$, $C \rightarrow D$) are chosen and their ending points exchanged: ($A \rightarrow D$, $C \rightarrow B$). The switching operation must meet two conditions: no bidirectional edges or self-loop. By applying this switching operation repeatedly on the input network, an ensemble of random networks is generated.

Backtracking. Backtracking[16] is a general method for finding all (or some) solutions to a computational problem, by incrementally building candidate solutions and then abandoning each partial candidate *c* (backtracking) as soon as it is determined that *c* cannot possibly be completed to a valid solution.

The backtracking algorithm enumerates a set of partial candidates that traverses this searching tree recursively, from the root down, in depth-first order. At each node *c*, the algorithm checks whether *c* can be completed to a valid solution. If it cannot, the whole sub-tree rooted at *c* is skipped (pruned). Otherwise, the algorithm (1) checks whether *c* itself is a valid solution, and if so reports it to the user and (2) recursively enumerates all sub-trees of *c*. The two tests and the children of each node are defined by user-given procedures.

We use it to enumerate all sub-graphs of a given size that occur in the input graph, because the backtracking algorithm keeps only the path of the initial and the end nodes, saving storage space and reducing space complexity. Then, we combine the way of enumerating all sub-graphs (backtracking) and the way of storing the input graph with an associated matrix that reflects the connection relationship of nodes and edges. The combination of the two algorithms can reduce the complexity of enumerating sub-graphs and searching time. Described in Figure 5, the detailed processes to enumerate sub-graphs are as below:

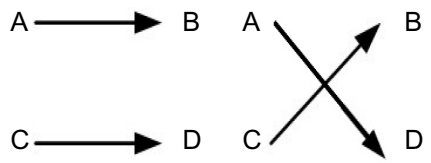


Figure 4. Schematic view of edge switching operator, edge replacement for generating random networks. As shown in this figure, the replacement process does not change the node degrees.

Carry out steps a–g sequentially to all nodes in the graph *G*.

a. Make two stack tables for all nodes $V_i \in V(G)$. Stack Table 1 stores the sub-graphs of a given size that contain the node V_i and are represented by their nodes and edges. Stack Table 2 stores all candidate nodes and edges that meet the conditions. The candidate nodes and edges are stored in pairs in stack Table 2.
b. Put the node V_i into stack Table 1, and then put all nodes and edges connected with V_i into stack Table 2, considering them as the candidate ensemble.
c. When the number of nodes in stack Table 1 meets the requirements, output the nodes and edges in stack Table 1 as a sub-graph.
d. Classify the sub-graph outputted by a different edge number.
e. If stack Table 2 is not empty, perform steps f–g.
f. When the number of the nodes in stack Table 1 is smaller than the designated size, remove the first candidate node and edge pair from stack Table 2, and put all nodes and edges connected with the removed node into stack Table 2 as the candidate nodes and edges (excluding the nodes in stack Table 1). If the node removed from stack Table 2 is the same as a node in stack Table 1, put the edge removed from stack Table 2 into stack Table 1 only; otherwise, put the node and edge removed from stack Table 2 into stack Table 1 at the same time, and then turn to step c to continue.
g. When the number of nodes in stack Table 1 is equal to the designated size, search the remainder nodes and edges in stack Table 2. If there is a node in stack Table 2 equal to a node in stack Table 1, put the edge corresponding to the node in stack Table 2 into stack Table 1, and then output the nodes and the edges in stack Table 1 as a sub-graph.

Remove the node V_i and all edges connected with V_i from the graph *G*.

Using the above steps, we can enumerate all sub-graphs through the backtracking algorithm and classify them based on the difference between the number of nodes and edges. This can reduce the space of isomorphism by judging the size of sub-graph beforehand. The combination of backtracking algorithm and associated matrix makes it easier and faster to search all sub-graphs than that with other algorithms. We provide the pseudo-code to show the process to enumerate sub-graph synoptically.

IInput: A graph $G = (V, E)$ and a given size n
Output: All *size-n* sub-graphs in G
for each node $v \in V$ **do**
$V_{Candidate} \leftarrow$(e,v′) *connect with v* and make a sub-graph set of *size-n SG*
Call **RecursiveFunction**(SG, $V_{Candidate}$, n)
RecursiveFunction (SG, $V_{Candidate}$, n)

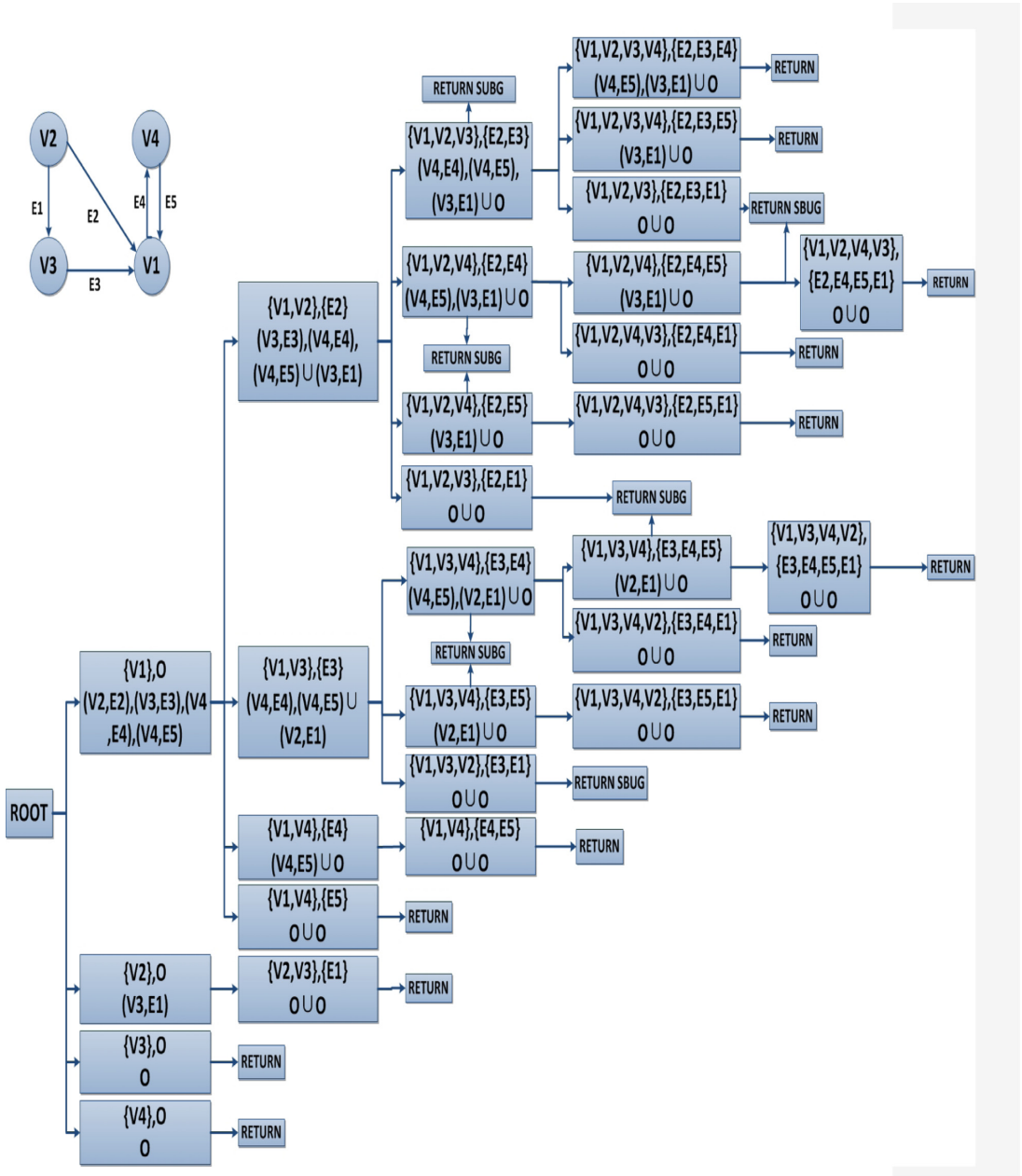


Figure 5. An example of searching sub-graphs by backtracking.

If $N_{SG}=n$ **then** output *SG* and **return**
While $(HeadPtr \in V_{subgraph})! = null$ **do**
If $N_{SG} < n$ **then**
If $HeadPtr->v' \in SG$ **then** $SG \leftarrow HeadPtr->e$
Else $V_{Candidate} \leftarrow (HeadPtr->e, HeadPtr->v') \cup SG \leftarrow (HeadPtr->e, HeadPtr->v')$ **call RecursiveFunction**(SG, $V_{Candidate}$, n)
else if $N_{SG}=n$ **then**
if $HeadPtr->v' \in SG$ **then** $SG \leftarrow HeadPtr->e$ and output SG
return

Time complexity for backtracking is $O(N_v * Ev_i * [\sum_{k=1}^{Size\_k} k + N_v * (Size\_k-1)])$. N_v is the number of nodes in the real network, Ev_i is the number of edges connected with v_i, $Ev_i = Out_{degree} + In_{degree}$, Size_*k* are the sub-graphs with a given size that we search in the real network. From the formula, the time complexity is associated with N_v, Ev_i, and Size_*k*.

Sub-graph isomorphism. As it is well known, among the different types of graph-matching algorithms, the sub-graph isomorphism is an NP-complete question.[14] So, time requirements of brute force matching algorithms (either in cases of isomorphism or sub-graph isomorphism) increase exponentially with the size of the input sub-graphs, restricting the applicability of graph-based techniques to problems implying sub-graphs with a small number of nodes and edges. We deal with the problem of sub-graph isomorphism by standardizing the associated matrix to mark the isomorphism class uniquely.

As previously stated, we standardize all sub-graphs to uniqueness. One way to identify isomorphic sub-graphs is to give an identical associated matrix to every isomorphic kind. For a sub-graph with N nodes, the number of permutations of the node label is $N!$. Meanwhile, there is a difficult problem about storing a digraph by using an associated matrix, because an element represents both connection relationship and orientation directly, especially "–1." So, a symmetric ternary sequence can be obtained by extracting the matrix row by row, and the symmetric ternary sequence is defined as *Code*; the code is unique for each isomorphic class. One *Code* marks an isomorphic class. We just count the number of *Codes* with the same value to detect the motif. Now we will confirm the sufficiency and necessity of this unique mark from the theory. The detailed processes are described as follows:

The sufficiency: If sub-graphs are isomorphic, their matrixes are similar. According to the properties of elementary changes of matrix, similar matrixes will become the same matrix after elementary changes; hence, *Code* value is the same.

The necessity: When the sizes (the number of rows and columns of the matrix) of two sub-graphs are equal, and both the numbers of out-degree and in-degree are equal in these matrixes ie, if these two matrixes have the same *Code* value, then the corresponding symmetrical ternary is the same, and also all elements of these matrixes in the corresponding position are same. Hence, these matrixes are identical.

The sufficiency and necessity are proved.

Evaluation measures. By using the results of the last step, the significance of each sub-graph found in the input and random networks is calculated. Here, a network motif is defined as a frequently or uniquely represented sub-graph in an input network; each motif must meet the measurement principles.

Frequency. This is the simplest measurement of estimating the significance of a motif. For a given network, the frequency is defined as the number of an isomorphism class occurrence $f(G)$ in the input network. It is more than 4.

P-value. This measure indicates the probability of a random network in which a motif $f(G)$ occurred no less than in the input network. Therefore, P-value ranges from 0 to 1. The smaller the P-value, the more significant is the motif.

$$P\text{-value} < 0.01$$

The last. If set N_{real} is the number of sub-graphs in the input graph and N_{rand} is the mathematical expectation of the number of random graphs, the probability of a random network in which a motif $f(G)$ occurred is no less than in the input network. A motif must meet this formula:

$$N_{real} - N_{rand} > 0.1N_{rand}$$

Each motif must meet the above principles, Motif = (Frequency, P-value, and the last). After all motifs are recognized, Z-score is a qualitative measure of statistical significance, which is defined as below. It indicates how important a motif is in the network.

$$Z_{score}(f(G)) = \frac{N_{real} - \bar{N}_{rand}}{\text{std}(< N_{rand} >)}$$

N_{real} is the number of $f(G)$ occurring in the input network, $\bar{N}_{rand}$ is the mean number at which $f(G)$ occurred in random networks, and std($< N_{rand} >$)is the standard deviation of the number of sub-graphs in the random networks. The larger the Z-score, the more significant is the motif, please refer to the [1].

Result and Discussion

In this section, we present the results of applying our algorithm to some real network; please see http://202.199.159.247/DownloadShow.asp?ID=15 for detail. Network instances applied are both biological and non-biological. The metabolic pathway of the bacteria *E. coli*, the transcription network of yeast, a sea urchin network, and an electronic network was targeted. These instances for testing the algorithm are up-to-date versions of the motif detection tests used by other existing algorithms. They can be freely obtained online at http://www.weizmann.ac.il/UriAlon/groupNetworksData.html.

The validity of algorithm. The numbers of motifs with different sizes observed in each network are presented in Table 1. In this table, we enumerate different kinds of networks with different numbers of edges, and calculate the number of some sub-graphs that occurred in the real network and the only *Code* that marks them.

From the first table, we know the different numbers of edges and nodes embody the difference between networks. Most motifs are similar in different networks, but small differences exist. The more similar the topology structures of motifs, the more similar are their *Code* values. This proves the accuracy of our unique marking of isomorphic sub-graphs.

In this table, the fifth column has the unique *Code* to represent the isomorphic class, the fourth column has the number of motifs in the real network, and the sixth column has the topological structures of motifs.

Algorithm correctness and extensive applicability. As is known to all, the Z-score is a parameter to measure the importance of the motif. The higher the Z-score, the more important the motif is. In Table 2, we compare our method and the ESA method.[1]

In this table, the second column has the topological structures of motifs, the third column has the names of motifs, the next column has the Z-scores of the ESA

Table 1. A summary of *Code* and the topological structure in different networks.

NETWORK	NODES	EDGES	SIZE-K	CODE	MOTIF
E coli	423	519	3	8528	
			4	12925664	
				236808	
S208	122	189	3	5668	
			4	13698880	
				12925664	
S420	252	399	3	5668	
			4	12925664	
				13698880	
S838	512	819	3	5668	
			4	12925664	
				13698880	

Table 2. The summary of *Z*-score in different networks, compared with the ESA method.

NETWORK	MOTIF	NAMED	Z-SCORE	
			ESA	OUR METHOD
E coli		Feed-forward loop	10	12.19
		Bi-fan	13	13.82
S208		Three-node feedback loop	9	11.51
		Four-node feedback loop	5	5.56
		Bi-fan	3.8	4.12
S420		Three-node feedback loop	18	18.91
		Four-node feedback loop	11	14.92
		Bi-fan	10	10.95
S838		Three-node feedback loop	38	40.05
		Four-node feedback loop	25	20.21
		Bi-fan	20	20.40

method,[1] and the final column has the *Z*-score of the method presented here.

Z-score is used to evaluate the importance of a motif in the network. In Table 2, the *Z*-scores of the same motif in different networks are different. For example, the *Bi-fan* motif has a *Z*-score of 13.82 in the biological network, 6.05 in the electronic circuits network (*S208*), 10.95 in the *S420* network, and 20.40 in the *S838* network. This shows that the importance of the same topological structure of motif in different networks is different. The higher the *Z*-score, the more important the motif is. So the *Bi-fan* motif is most important in the *S838* network. In addition, the *Z*-scores of different motifs in the same network are different. In *S838*, the *Z*-score of the *three-node feedback loop* motif is the largest. So, it is most important and occurs more frequently than other motifs.

Meanwhile, in our method, the topological structures of motifs are the same as in the ESA method.[1] The importance of different motifs in the same network and that of the same motif in different networks is similar to that in Milo et al.[1] This proves the correctness of our method. In addition, our method achieves mining of motifs in different kinds of networks, which proves our method is valid and offers extensive applicability.

The comparison of accuracy. We have proved the validity and correctness of our method – Edges-Nodes Search Algorithm (ENSA), and will below prove its accuracy and comparison with that of Feature Compression for Graph

Table 3. The comparison of accuracy.

SIZE(N)	TYPE	FCGI	ACCURACY	ENSA	ACCURACY
3	13	13	100%	13	100%
4	199	199	100%	199	100%
5	9364	9364	100%	9364	100%

Isomorphism (FCGI).[17] In this table, Type is the number of non-isomorphic graphs with size n that are discerned by our method (Table 3).

We use the backtracking algorithm to enumerate all sub-graphs of a given size that occur in the input graph. Our method is mining a non-probabilistic and exact motif in the network. So, it is an inherent property of the network that the number of types with size n is same as in the network. But the backtracking algorithm just keeps the path of the initial and the end nodes, which saves the storage space and reduces the space complexity. Then, we combine it with the associated matrix, which reflects the connection relationship of nodes and edges. The combination of the two algorithms can reduce the complexity of enumerating sub-graphs and searching time.

The comparison of searching time. In order to judge the performance of our method, we carried out a series of experiments on a set of testing data. The performance evaluation measured the amount of time consumed. We enumerate all sizes of sub-graphs for testing, and then compare with method FCGI.[17] The operating environment was consistent, as all evaluation was performed on the same computer hardware and software: Intel® Core™ 2 Quad CPU, 2.4 GHZ work station, 3 G RAM, Windows 7 operating system, and Visual C++ 6.0. This ensures that the comparison is meaningful (Fig. 6).

Searching the sub-graphs in the network is the key part in the process of recognizing motifs. The efficiency of enumerating all the sub-graphs is an important standard to measure the quality of the algorithm, because it costs much time to search the sub-graphs in the whole process. According to the previous research, ESA's search time is the longest, ESU takes the second longest, and ESU-Tree is the shortest. The work of Hu et al enumerates all the sub-graphs by the ESU-Tree algorithm.[17] In comparing our method with this use of ESU-Tree, we found only a small difference in the time required to enumerate the sub-graphs. In cases with three and four nodes, our backtracking algorithm took just 10^{-2} and 10^{-1} seconds longer. This gap can be reasonably ignored, and they can be regarded as equal. In cases with five nodes, the time spent on enumerating the sub-graphs by the backtracking algorithm is shorter than by ESU-Tree, by a significant 27.5 seconds. This is a relatively big improvement. The shorter time spent on enumerating reflects the superiority of the algorithm presented here.

The time complexity of our algorithm is $O(N_v * Ev_i * [\sum_{k=1}^{Size\_k} k + N_v * (Size\_k - 1)])$.

The time complexity of the FCGI algorithm is $O(N_v * Vv_i * [(Size\_k - 1)^2 * Vv_i + Size\_k^2])$.

N_v is the number of nodes in the real network; Ev_i is the number of edges connected with v_i; $Ev_i = Out_{degree} + In_{degree}$, $(0 \le Ev_i \le 2(N_v - 1))$; and Vv_i is the number of nodes connected with v_i, $0 \le Vv_i \le Nv - 1$. Size_k are the sub-graphs with a given size that we search in the real network. In the yeast network, when we searched the sub-graphs with Size_k = 3, the largest frequentness are $N_v * 2(N_v - 1) * 1382$ in our algorithm and $N_v * (N_v - 1) * 2757$ in FCGI. This gap can be ignored, and they can be regarded as equal. But for Size_k = 5, the

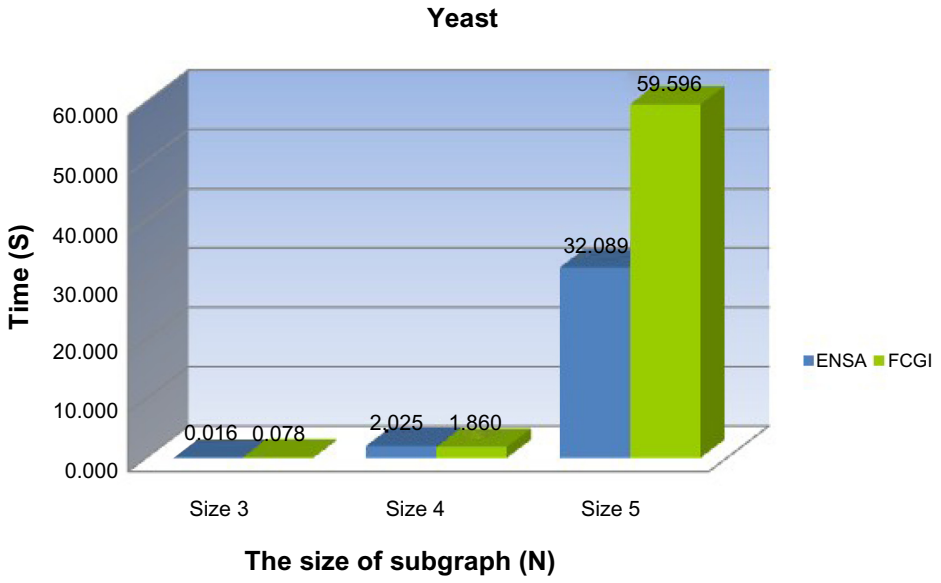


Figure 6. The performance evaluation is reflected in consuming time, using the *yeast* network as the testing data. The blue column denoting the time consumed in our algorithm is compared with the green column, which represents time using the method of the ESU-Tree.[17] The *x*-coordinate denotes the size of sub-graph, and the *y*-coordinate denotes the searching time.

largest frequentness probabilities are $N_v * 2(N_v - 1) * 2767$ and $N_v * (N_v - 1) * 11{,}017$, for the backtracking and FCGI algorithms, respectively. The frequentness of the backtracking algorithm is less than two times that of *FCGI*. This proves that the comparison of the searching time is valid, and the backtracking method is superior.

Because of the limitations of the algorithm FCGI, we only tested and compared the yeast network. But our method has more extensive applicability; hence, we list the searching times of enumerating some sub-graphs in the other networks, as a reference. The results are given in Table 4.

We list the searching time in the different networks with different sizes in this table. In any network, the searching time is different when the sizes of sub-graphs are different. For example, in the sea urchin network, 0.003 seconds where there are three nodes in the sub-graph, 0.012 seconds for four nodes, and 0.237 seconds for five nodes. With the size increasing, the search time is amplified 10-fold. It shows that the relationship between the searching time and the size of sub-graph is directly proportional. Among the networks we have tested, the searching time was the shortest in the *S208* network, in which we achieved enumeration of eight-node sub-graphs. The shortest time is at the microsecond level, and the longest is 20.326 seconds.

Conclusion

In this paper, we presented a method for mining network motifs. Based on the associated matrix, all sub-graphs could be enumerated by adding edges and nodes progressively with our backtracking algorithm, which saves storage space and reduces space complexity because it keeps only the path of the initial and the end nodes. This way of storing an input graph by an associated matrix reflects the connecting relationship of nodes and edges. Taking advantage of the combination of the associated matrix and the backtracking algorithm, our method reduces the complexity of enumerating sub-graphs, providing a more efficient solution for enumerating sub-graphs during motif mining. Then, the associated matrix is standardized and the isomorphism sub-graphs are marked uniquely by using the symmetric ternary to translate the elements (–1,0,1) in the associated matrix to a decimal number Code, which represents an isomorphism class. We judge whether the two sub-graphs are isomorphic by simply checking whether the Codes are the same or not. Because the associated matrix is usually not a square matrix and "–1" reflects direction, it is difficult to deal with them. Our method gives a new application of the associated matrix in the digraph domain by combining it with the symmetric ternary. It is easier and faster than other algorithms. In addition, from the results obtained, we have proved the correctness and effectiveness of our method. The comparison of the results of our method with the ones obtained shows that our approach has lower searching time and more extensive applicability.

Table 4. Search time in different networks. The first column has the names of different networks, and columns 2–7 has the sizes of sub-graphs and time.

TIME(S) NETWORKS	SIZE 3	SIZE 4	SIZE 5	SIZE 6	SIZE7	SIZE 8
E. coli	0.053	0.556	10.093			
Sea urchin	0.003	0.026	0.237	2.794		
S208	0.003	0.012	0.05	0.181	0.825	4.157
S420	0.011	0.044	0.142	0.692	3.648	20.326
S838	0.037	0.122	0.554	3.13	18.696	

Acknowledgment

We would like to thank the anonymous reviewers for helpful comments.

Author Contributions

Conceived and designed the experiments: YX, QZ. Performed the experiments: YX, CZ. Analyzed the data: QZ. Contributed reagents materials/analysis tools: YX, CZ. Wrote the paper: YX. All authors reviewed and approved of the final manuscript.

REFERENCES

1. Milo R, Shen-Orr S, Itzkovita S, Kastan N, Chklovskii D, Alon U. Network motifs: simple building blocks of complex networks. *Science*. 2002;298(5594):824–7.
2. Koyutürk M, Subramaniam S, Grama A. Introduction to network biology. In: Koyutürk M, Subramaniam S, Grama A, eds. *Functional Coherence of Molecular Networks in Bioinformatics*. New York: Springer; 2011:1–13.
3. Alon U. Network motifs: theory and experimental approaches. *Nat Rev Genet*. 2007;8:450–61.
4. Kashtan N, Itzkovitz S, Milo R, Alon U. Efficient sampling algorithm for estimating subgraph concentrations and detecting network motifs. *Bioinformatics*. 2004;20(11):1746–58.
5. Wernicke S. Efficient detection of network motifs. *IEEE/ACM Trans Comput Biol Bioinform*. 2006;3(4):347–59.
6. Itzkovitz S, Milo R, Kastan N, Ziv G, Alon U. Subgraphs in random networks. *Phys Rev E Stat Nonlin Soft Matter Phys*. 2003;68(2):1–8.
7. Tian LJ, Liu CQ, Xie JQ. A partition method for graph isomorphism. *Phys Procedia*. 2012;25:1761–8.
8. Ding H, Huang Z. Isomorphism identification of graphs: Especially for the graphs of kinematic chains. *Mech Mach Theory*. 2009;44(1):122–39.
9. Knossow D, Sharma A, Mateus D, Horaud R. Inexact matching of large and sparse graphs using Laplacian eigenvectors. *Pattern Recognit*. 2009;5534:144–53.
10. Ogras UY, Marculescu R. Energy-and performance-driven NoC communication architecture synthesis using a decomposition approach. *Proceedings of the Conference on Design, Automation and Test in Europe*. Vol 1.Washington, DC; 2005:352–7.
11. Batagelj V, Mrvar A. Pajek – analysis and visualization of large networks. In: Jünger M, Mutzel P, eds. *Graph Drawing Software*. Berlin: Springer Verlag; 2004:77–103.
12. Schreiber F, Schwöbbermeyer H. MAVisto: a tool for the exploration of network motifs. *Bioinformatics*. 2005;21(17):3572–4.
13. Wernicke S, Rasche F. FANMOD: a tool for fast network motif detection. *Bioinformatics*. 2006;22(9):1152–3.
14. Bondy A, Murty USR. *Graph Theory*. Berlin: Springer Verlag; 2008.
15. Milo R, Kastan N, Itzkovitz S, Newman MEJ, Alon U. Uniform generation of random graphs with arbitrary degree sequences. *arXiv:cond-mat/0312028*. 2003;106:1–4.
16. Xie P. A dynamic model for processive transcription elongation and backtracking long pauses by multisubunit RNA polymerases. *Proteins*. 2012;80(8):2020–4.
17. Hu J, Sun L, Yu L. A novel graph isomorphism algorithm based on feature selection in Network Motif Discovery [OL]. *Science Paper Online*. 2011. Available at http://www.paper.edu.cn/en_releasepaper/content/4441016