Application of a K-Ladder Connectivity Algorithm for Clustering of Protein Evolutionary Network

Reshma Nibhani, Avi Soffer, Ahuva Mu'alem, Zeev Volkovich, and Zakharia Frenkel

***Abstract*—An evolutionary network (EN) in formatted protein sequence space is a very large graph representing information about sequence similarity of relatively short protein fragments. This graph can be used for detecting hidden relatedness between proteins, which is highly significant in protein annotation. Effective EN analysis requires an appropriate graph clustering approach. Based on the fact that biological relatedness is strongly dependent on the number of independent graph nodes connections, we develop a network clustering method that is capable to produce quality clusters the members of which have a satisfactory level of relatedness.**

In this article we describe a new network partitioning method which is based on the k-cycles graph connectivity approach. After formally defining a unique structure, named k-ladder connectivity, we demonstrate that the k-ladder-based algorithm is able to successfully detect the groups of functionally related proteins.

To exhibit the quality of the method, we have conducted a set of experiments in which it has been very effective in clustering of EN, as well as the significantly denser protein-protein interaction networks (PPINs). Furthermore, it can be simply adapted for more complicated structures than cycles, as well as applied to other large networks of different types.

***Index Terms*—K-ladder, connectivity algorithm, network clustering, protein evolutionary network, formatted protein sequence space, protein-protein interaction networks.**

I. Introduction

Proteins are the main components in all living organisms. Significant progress in molecular genetic technology during the last decade provided us with a vast amount of protein sequences that exist in nature. For example, the recent release of the UniProt database (http://www.uniprot.org/) contains more than 40,000,000 protein sequences. However, many of these protein sequences have no proper annotation – meaning that the structure and biological function of the corresponding proteins are unknown. Such characterization of these proteins on the basis of their known sequences and often according to some other high-throughput information is one of the main challenges in computational biology.

Among a multitude of bioinformatics methods and algorithms dedicated to reveal the sought-after protein organization and biological functionality, there is a group of approaches that use graph analysis techniques applied to various kinds of protein associated networks. A common example of such a network is the Protein-Protein Interaction network (PPI or PPIN), a dedicated graph used for integrative representation of the structure and behavior of biological systems.

The Protein-Protein Interaction network is modeled as an undirected graph $G= (V, E)$, which may include a large number of components. V is a set of nodes representing mainly proteins, but may also represent genes, RNAs and other molecules. E is a set of edges representing protein pairwise interactions. The definition of the edges is method specific: in some models the edges represent physical interaction or co-membership in protein complex. Edges may also carry weights to represent the probability of an interaction.

PPINs are successfully used for prediction of a single protein function, signaling and metabolic pathways, as well as to understanding of physiological processes and molecular basis of some diseases. Clustering of PPINs is aimed at identifying two types of cellular modules: protein complexes and functional modules [1]. Protein complexes are groups of proteins that interact with each other at the same time and place, forming a unique multi-molecular machine. Functional modules, in contrast, consist of proteins that participate in a particular cellular process while binding each other at various conditions or phases of the cell cycle. This aspect of PPINs is very essential for functional annotation of proteins.

Another type of protein network, which can be used for the functional annotation, is Protein Evolutionary Network (EN) [2]-[4]. The EN is a graph formed by proteins sequence fragments of a specific size (about 20 amino acids). The connection condition in this network is characterized by a small Hamming distance (which means high sequence identity). The most important property of the EN is the existence of paths in which protein sequences may gradually change from a particular initial sequence to a completely different one, while conserving the structural and functional properties of the corresponding protein fragments [3]-[4]. This phenomenon enables finding hidden relatedness between proteins, which cannot be detected by other methods. This knowledge is very important for protein annotation [3], [5] and studies of evolutionary processes [6].

In the case of EN, there are at least two reasons for the application of clustering procedure. The first one is associated with the fact that the clusters can detect groups of functional or structural relatedness [5], [7], which can be used for protein classification and sequence annotation. The second is that the clustering procedure can potentially help to reduce the need for storage, and can also ease the access to ENs of very large size. For example, in a protein sequence database of 10^7 sequences there are more than 10^9 fragments of 20 amino acids. This means that EN of such database would consist from 10^9 nodes, which can have about 10^{18} edges! Although real ENs are far from being a clique, the

Manuscript received May 14, 2014; revised July 19, 2014. This work was supported by the European Union seventh framework program via the PathoSys Project (grant number 260429).

The authors are with the ORT Braude College of Engineering, Karmiel, Israel and Research Fellow at Institute of Evolution, University of Haifa, Israel (e-mail: reshma.iidsalld2007@gmail.com, asoffer@braude.ac.il and ahumu@yahoo.com, vlvolkov@braude.ac.il, zakharf@research.haifa.ac.il).

storage requirements, of all connections, for a full protein universe is practically enormous. A simple way to overcome this problem is to combine nodes in groups (clusters) and to refer to these clusters as super-nodes. This would substantially decrease the number of connections, as is illustrated in Fig. 1. This idea is somewhat similar to the well-known multilevel approach [8], with the difference that in our case the super-nodes are groups of some functional or structural relatedness. Another requirement from the algorithm is that it should be fast enough and scalable, to make possible its application during the EN creation, where the calculation speed is a crucial parameter.

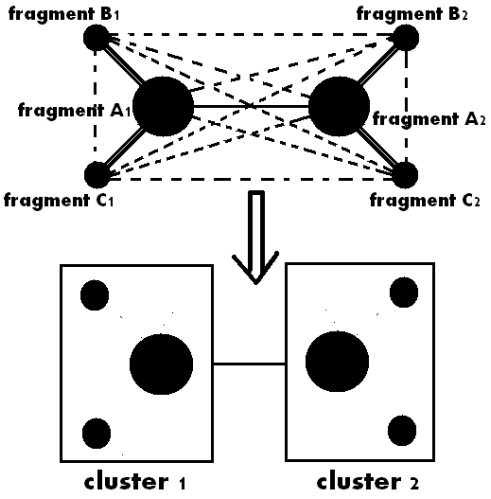


Fig. 1. Combining nodes in groups (clusters) substantially decreases the amount of connections.

There is a various graph clustering methods which are applied for the analysis of biological, in particular PPI networks (see [1], [9], [10]). However, there is no universal approach which can be satisfactory for all cases. Each specific network and corresponding clustering procedure requires application of the approach with best match to this specific need. In many cases the clusters are defined as groups with increased amount of internal connections – in comparison to external (i.e. cluster selection is a type optimization problem). In other cases the clusters are defined as groups with some specific properties (without comparison with the rest part of graph). Similarly, the networks can be of various sizes, densities and models (i.e. random, scale-free and so on).

The challenge that motivated us in this work was to develop a new clustering method that focuses on the connected components and works well for ENs. In addition, we propose several approaches for the adaptation of our technique to smaller but significantly more dense PPINs.

The rest of this article is organized in the following way:

In Section II we present and explain the methods and algorithms developed. In Section III we describe the set of experiments conducted in order to test the ability of our methodology to partition a protein network. We also present and discuss the results obtained in these experiments. We conclude with a general discussion and an outlook in Section V.

II. Methodology

In this section we describe and explain the new concepts, methods and algorithms we developed in order to properly partition EN and more.

A. Approach

The *K-ladder connectivity* is a new approach to graph clustering, that we propose as an alternative to other commonly used approaches to node-connectivity, such as the bi- and tri-connected components [11], [12] . According to the K-ladder basic approach, two cycles (each composed by less or equal to K edges) are considered connected if and only if they have at least one common edge. In contrast to the classic connectivity approach, the proposed method deals with certain structural elements connections rather than simple node connections. The connectivity approach is selected for EN analysis is due to the fact that similarity of protein fragments is strongly dependent on the number of independent pathways (e.g., maximum flow) between fragments [5]. The classic connectivity approach has a certain limitation, since as clusters become larger, the probability of the appearance of new pathways between nodes increases dramatically. This fact causes a performance decrease. It is shown in this work, that the application of a well-known decomposition of a graph into bi- and tri-connected components, according to [11], [12], does not work effectively for this case. In contrast, the K-ladder connectivity-based algorithm, proposed here, provides satisfactory results. In addition, we enhance the K-ladder connectivity definition, by providing stronger requirements for connectivity, in order to reduce noise caused by uncertain connections.

B. Foundation of Ladder Path and K-Ladder Connected Component

In this section we explain and formalize the K-ladder connectivity graph partitioning concept.

We design an algorithm to compute the ladder-connected components of a graph and formally show the correctness of this algorithm. We start with some basic definitions.

An undirected *graph* $G=(V,E)$ is a pair, where V is a set of nodes and E is a set of 2-element subsets of V, called edges. The degree of a node $v\in V$ denoted $\deg(v)$ is the number of edges incident to the node. A *path* in G is a sequence $p=(v_1,v_2,...v_t)$ such that $(v_i,v_{i+1})\in E$ for all $i\in\{1,....,t-1\}$. A path $p=(v_1,v_2,...v_t)$ is called *simple* if $v_i\neq v_j$ for all $i\neq j\in\{1,...,t\}$. $v\in p$ denotes that v is a node in the path p . Similarly, $e\in p$ denotes that e is an edge in the path p . The *size* of a path is the number of distinct edges in the path. A *cycle* in G is a path $c=(v_1,v_2,...v_t)$ with $v_1=v_t$. A cycle $c=(v_1,v_2,...v_t)$ is called simple if $v_i\neq v_j$ for all $i\neq j\in\{1,...,t\}$. We say that two cycles c and c' are *edge-intersecting* if they share a common edge (i.e., if $\exists e\in E$ such that $e\in c$ and $e\in c'$). We now formally define the notion of a ladder path between two nodes.

Definition: We say that the edges e and e' are k *-ladder connected* if either $e=e'$ or if there exists a finite sequence

of cycles $c_1, c_2, ..., c_d$) in G such that:

$e \in c_1$ and $e' \in c_d$.

Each size of the cycle c_i is at most k.

Each cycle c_i in the sequence is simple.

Each consecutive cycles c_i and c_{i+1} in the sequence are edge intersecting.

An example 3-ladder connection is given in Fig. 2.

Fig. 2. Example of ladder connection. The *edges* (a,b) and (x,w) are 3-ladder connected. However, the *edges* (a,b) and (y,z) are not 3-ladder connected, since the size of the cycle (x,y,w,z) is 4.

To define the k-ladder connected components of G we first need to show that the ladder connectivity forms an equivalence relation. More formally, we denote $e \sim e'$ if and only if the edges e and e' are k-ladder connected in G, and show that it is an equivalence relation. This equivalence relation induces a natural partition of the edges of the graph G into disjoint subsets, called the k-ladder connected components of G.

Claim: The relation $\sim$ is an equivalence relation.

Proof: Reflexivity: observe that $e \sim e$ by definition. Symmetry: if $e \sim e'$ then $e' \sim e$ by the fact that G, is an undirected graph, using the same sequence of cycles connecting e to e' in a reverse order. Transitivity: If $e_1 \sim e_2$ and $e_2 \sim e_3$, then by concatenating the sequence of cycles connecting e_1 to e_2 with the sequence of cycles connecting e_2 to e_3 (through the common edge e_2) we get that $e_1 \sim e_3$.

We now define a new graph $G^* = (V^*, E^*)$ corresponding to the given graph $G = (V, E)$. We will later use G^* to compute the k-ladder-connected components of G. From now on we shall assume that $G = (V, E)$ and k are fixed.

Definition: Let the node set of the undirected graph $G^* = (V^*, E^*)$ be $V^* = \{v_{(x,y)} \mid (x,y) \in E\}$. Let the edge set be $E^* = \{v_{(x,y)}, v_{(y,z)})$ | There exists a simple cycle $c \in G$ with size at most k, where $(x,y) \in c$ and $(y,z) \in c\}$.

Intuitively, $G^* = (V^*, E^*)$ has a node corresponding to each edge of G Additionally, $G^* = (V^*, E^*)$ has an edge joining two neighboring edges of G if the two neighboring edges lie on a short simple cycle in G. An example for a graph G and its corresponding graph G^* for $k = 3$ is given in Fig. 3.

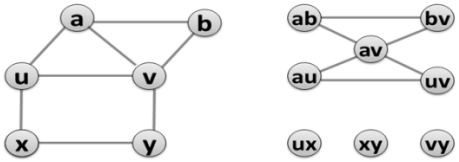


Fig. 3. Example for a graph G and its corresponding G^* (for $k = 3$) Edges (a,b) and (u,v) are 3-ladder connected in G. Observe that the nodes $v_{(a,b)}$ and $v_{(u,v)}$ are on the same connected component of G^*. Conversely, (a,b) and (x,y) are not 3-ladder connected in G. Observe that $v_{(x,y)}$ is an isolated node in G^*, and in particular $v_{(a,b)}$ and $v_{(x,y)}$ are not in the same connected component of G^*.

We now define the following standard equivalence relation on the *nodes* of G^*. We denote $v_{(x,y)} \approx v_{(a,b)}$ if and only if there exists a path from $v_{(x,y)}$ to $v_{(a,b)}$ in G^*.We next show that the k-ladder components of G are exactly the connected component of G^* Therefore the problem of finding the k-ladder components of G reduces to the problem of finding the connected components of G^*. More formally:

Proposition: $(x,y) \sim (a,b)$ if and only if $v_{(x,y)} \approx v_{(a,b)}$.

Proof: If (x,y) and (a,b) are k-ladder path connected in $G = (V, E)$, then there exists edge-intersecting cycles $c_1, c_2, ..., c_d$ each with at most k edges where $(x,y) \in c_1$ and suppose $e_i = (u_i, v_i) \in c_i$ and $e_i = (u_i, v_i) \in c_{i+1}$.

Clearly, there is a path $p = (x, y, ..., u_1, v_1, u_1, ..., u_{d-1}, v_{d-1}, u_{d-1}, ..., a, b)$ in G starting from node y and ending at b along the edges of $c_1, c_2, ..., c_d$. Observe that any two neighboring edges on the path p belong to a simple cycle with at most k edges and thus $v_{(x,y)} \approx v_{(a,b)}$.

Conversely, if there is a path $p' = (v_{(a,b)}, v_{(b,c)}, ..., v_{(x,y)})$ from $v_{(a,b)}$ to $v_{(x,y)}$ in G^*, then any two neighboring nodes in p' represent neighboring edges on a cycle with at most k edges in *G*. One can use these cycles to show that (a,b) and (x,y) are k-ladder connected in G.

The correctness of the next algorithm immediately follows from the above proposition.

Notation: The graph $G \setminus \{v\}$ is the graph defined by removing the node v from the graph G.

C. Methods

In this section we present two methods for graph decomposition, using k -ladder connectivity analysis. In section (III) we present a third k -ladder-based approach, which includes an enhancement to the basic k -ladder connectivity definition. The motivation for developing the enhanced approach is explained at the review of the research experiments.

1) Structured decomposition algorithm using G*

First we formalize the graph partitioning method, which uses the basic algorithm for detecting the k -ladder connectivity components. This method builds the G* graph that is defined in the previous Section (II. 2).

Decomposition Algorithm via G^*:

Input: undirected graph $G=(V,E)$ and an integer $k>0$.

Output: the k -ladder components of G.

$$V^* \leftarrow \phi, E^* \leftarrow \phi$$

For every (x,y) in E:

$$V^* \rightarrow V^* \bigcup v_{(x,y)}$$

For every (u,v) in E:

If $\deg(u)=1$ or $\deg(v)=1$,

Continue to the next edge.

Run $BFS(G\setminus\{v\})$ using node u as a starting node and compute

The distance $d_{G\setminus\{v\}}(u,x)$ from u to every node x in the graph $G\setminus\{v\}$.

This BFS should be carried out only up to breadth $k-2$.

For every $(v,w)\in E$:

$$\text{If } d_{G\setminus\{v\}}(u,w)\le k-2$$

$$E^* \leftarrow E^* \bigcup (v_{(u,v)}, v_{(v,w)})$$

Run $DFS(G^*)$ to compute the connected components of $G^*=(V^*,E^*)$.

2) Optimized (heuristic) decomposition algorithm

The algorithm described above can be rather bulky if the network is sufficiently large and dense. For faster calculations, in case of simple k -ladder components, the following algorithm can be used:

Input: undirected graph $G=(V,E)$ and an integer $k>0$.

Output: the k -ladder components of G.

Create:

L_1– list of nodes for "local BFS" (3);

L_2– list of edges of a selected cycle;

L_3 – the first list of the connected components (integer). The size of the list is equal to current number of the connected components. Each member (i) of the list points to another member $(j, i\ge j)$, which means that component i was attached to j. In the beginning (i.e. when a new component (member) is added) $i=j$.

For each edge $k=(m,n)$ associate variable lk pointing to index of corresponding connected component in L_3. Initial value of each $lk=-1$ (i.e. non-associated).

L_4 and L_5 lists of indexes from L_3.

Run $BFS(G)$ starting from the some node u and compute the distance $d(u,w)$ in the graph G for every w. For every frond edge (v,w) in E put one node (with maximal d) in to L_1.

For every w from L_1 do "local BFS":

Make $BFS(G)$ starting from w (up to breadth $k/2+1$ for the even k and $(k+1)/2$ for odd k); in this BFS we don't open nodes v, such that $(u,v)>d(u,w)$;

For each opened node the correspondent "father" node is remembered;

Each frond edge (m,n) is corresponding to a simple cycle with at most k edges; to select this cycle do:

Clear L_2 and add (m,n) to L_2;

$v_1=m$;

while ($v_1\neq w$)

{

v_2 = father (v_1);

add (v_1,v_2) to L_2;

$v_1=v_2$;

}

$v_1=m$;

while ($v_1\neq w$)

{

v_2 = father (v_1);

if ((v_1,v_2) is already present in L_2)

delete (v_1,v_2) from L_2;

else

add (v_1,v_2) to L_2;

$v_1=v_2$;

}

After this procedure the list L_2 contains all edges of the selected cycle.

Associate each new selected cycle to one of the already selected connected components or create a new component:

if $lk=-1$ for every edge from L_2

add a new component to L_3;

$lk=sizeof(L_3)$ for every edge from L_2

else

for each pair of edges K_1 and K_2 from L_2 do

if $(lk_1=-1 \,\&\, lk_2\neq -1) lk_1=lk_2$

if $(lk_1\neq -1 \,\&\, lk_2\neq -1)$ unite connected components lk_1 and lk_2 (as described below)

Uniting the connected components lk_1 and lk_2:

Clear L_4 and $L_5\sqrt{b^2-4ac}$.

Add $lk1$ to L_4;

While ($L_3[lk_1] \neq lk_1$)

$lk_1 = l_3[lk_1]$;

add lk_1 to L_4;

add lk_2 to L_5

while ($L_3[lk_2] \neq lk_2$)

$lk_2 = l_3[lk_2]$;

add lk_2 to L_5;

For each member k from L_4 and L_5 do:

$$L_3[L_4[k]] = \min\{lk_1, lk_2\} \&$$

$$L_3[L_5[k]] = \min\{lk_1, lk_2\}$$

Renumbering and output of components

III. Results

We have conducted a set of experiments in order to test the ability of our methodology to partition various protein networks. In this section we describe these experiments and present and discuss their results.

In the first experiment we tested our k-ladder based algorithm on an EN. The network partitioning was done as a multi-step process: Step 1 includes preparations (or pre-processing) of the network, in order to find the major connectivity components and reduce the network's size. In the second step we compared bi-, tri- and then k-ladder connected components-based decompositions to the outcome of step 1. The k-ladder connectivity approach performed well and produced satisfactory results.

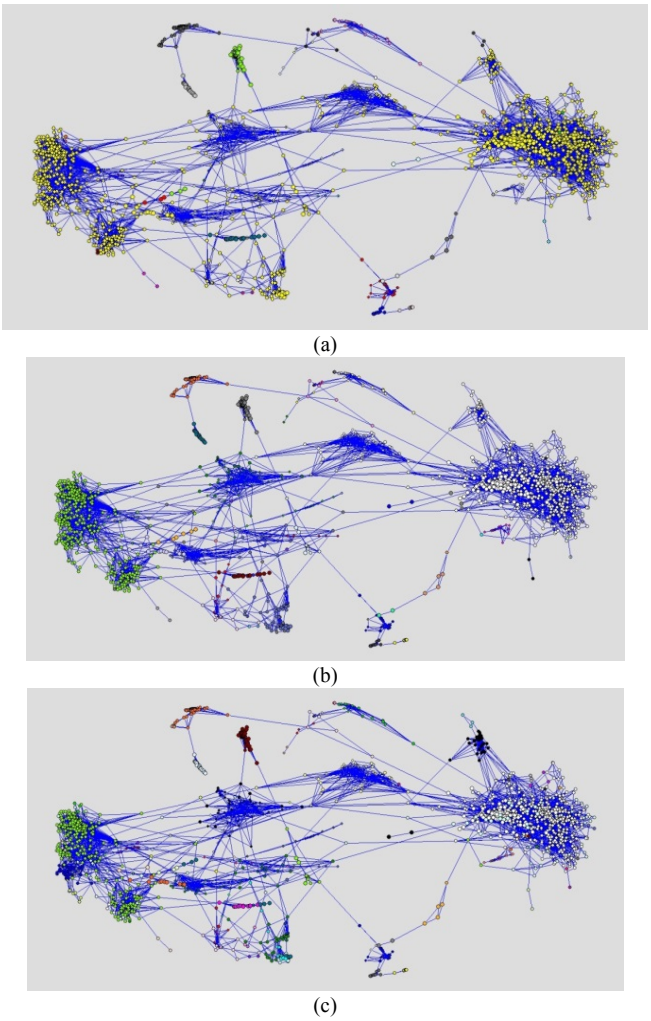
We then tried the basic k-ladder algorithm on PPI networks (which are denser). This experiment is described in as follows:

A. Step 1: EN Preprocessing

The EN used in this experiment is described in [2], and available at http://www.genome.haifa.ac.il/~zfrenkel/DATABASES1/. It is constructed from 112 prokaryotic proteomes (a total of about 32×10^4 proteins) and contains $5.98*10^7$ nodes (the "orphan" fragments were not taken into account) and 4.79×10^8 edges. To reduce the network size we selected the components which were connected at least in 80% sequence identity level (called "threshold"). These components were considered as super-nodes and all their external connections were preserved. The distribution of such components is described in [2]. This approach enables to dramatically decrease the average density of the network. The new reduced network contains 2.57×10^7 nodes (including the super-nodes) and 3.63×10^7 edges. That means the density of the network was decreased in more than five times by this procedure. In addition, only nodes concerning to 3-core were selected for further analysis. All nodes with degrees less than three were repeatedly deleted until the amount of nodes stopped changing. The resulting network contains 5.54×10^6 nodes and 1.87×10^7 edges. The largest connected component contains 1.08×10^6 nodes (about 20% of the graph). The second component contains only 2126 nodes (see Table I, (1)). These two main components were selected for further analysis.

B. Step 2: Bi-, Tri- and K-Ladder Connected Components

The two largest connected components obtained in the previous step were selected for further partitioning. As shown in Table I ((2)-(5)), the bi- and tri- connected components-based methods were able to detect only relatively small components, while the main part of graph stays undivided. However, as may be directly observed in the graph visualization done by Fruchterman-Reingold's force-directed algorithm [13] using the 3D mode of the Pajek visualization tool [14], this main part of graph is composed of several clusters, that were not detected by the tri-connected algorithm (Fig. 4A). In contrast, using the *3-ladder* connected components method allows to split it into compact meaningful clusters (as seen in Fig. 4 B, C). It is very important to note, that the partition obtained by Ladder-with-an-a *3-Ladder* connected components is compatible with the biological functions of the corresponding proteins (Fig. 5). As expected [5], the clusters connected by many alternative connections have relatively similar functions (in our case, different families of transcriptional regulators), whereas better separated clusters have more distinct functions.



(a)

(b)

(c)

Fig. 4. The Partitioning of the second largest component via selection of tri-connected components (a), 4-ladder connected components (b) and 3-ladder connected components (c). Partitions are shown by different colors.

The code for the selection of bi-connected and tri-connected components was taken from OGDF (Open Graph Drawing Framework) - a self-contained C++ class library, which can be found at http://www.ogdf.net/doku.php.

C. PPI Network Clustering

The yeast PPI network used in this paper is described in [15]. Application of the k-ladder algorithm, described above in its basic form, to this network was unable to decompose the main part of the graph and more than 9,000 nodes from the 12,000 are remaining in one cluster. For this task, several types of connectivity structures were tested, such as "cycle of cycles". We chose to use an extreme case of this connectivity definition, where the connected elements of the 'ladder' are considered cliques of a certain selected size, and the condition for their connection is the existence of a chosen amount of common nodes. Such an approach was suggested in [16] as percolation of cliques [16], [17].

TABLE I: RESULTS OF PARTITION (ORDERED BY SIZE). SIZES ONLY OF THE FIRST 10 LARGEST COMPONENTS ARE SHOWN

	(1)	(2)	(3)	(4)	(5)	(6)	(7)
1	1.07724E6	609091	433976	1586	1574	6403	561
2	2126	1750	1707	133	122	5861	347
3	2017	1740	1664	83	84	5555	116
4	1928	1557	1646	64	65	3423	103
5	1895	1555	1591	55	41	3131	96
6	1846	1517	1550	21	21	3058	93
7	1829	1405	1541	19	19	3054	75
8	1713	1390	1496	18	18	3021	64
9	1671	1380	1496	15	16	2963	47
10	1612	1379	1493	13	13	2954	41

(1) – connected components of the primary graph after "pre-clustering" and filtering;
(2), (3) – bi- and tri-connected components of the largest component of (1);
(4), (5) – bi- and tri-connected components of the second largest component of (1);
(6), (7) – 3-ladder connected components of the first two largest components of (1);

TABLE II: DECOMPOSITION OF THE PRE-PROCEEDED YEAST PPI NETWORK USING *3*-LADDER CONNECTIVITY ALGORITHM

Threshold	Cluster order number (ranged by size)									
	1	2	3	4	5	6	7	8	9	10
50	64	61	40	27	20	18	16	16	15	14
60	147	134	64	55	45	34	28	27	26	24
70	285	283	119	93	60	58	47	46	35	29
75	798	297	117	64	47	37	35	30	28	26
80	1409	166	48	42	41	40	31	30	25	24

A drawback of the "cliques" method is a very high level of overlapping of the clusters that makes it difficult to accurately interpret the results. In an attempt to overcome this drawback, we enhanced the basic percolation method by adding another step: each detected cluster is deleted from the network. This modification contributes to reduction of the cluster overlapping level, as well as gradually simplifying the graph complexity. This method produced better results (data not shown). Nevertheless, the weakness of this approach lies in the fact that the graph partitioning and reduction process (i.e. cluster deletion) is dependent on the details of the particular clustering algorithm which leads to unstable results.

In an attempt to stir away from this weakness we tried several other possibilities. For example, we tried to clean the noise by using the Jacquard index. I.e. for each edge if the amount of common neighbors of these nodes is less than some selected threshold, the edge was deleted. This approach does not work for PPIN: even for very high threshold, (more than 80 common nodes) the network was decomposed into multiple single nodes and one dense connected component, which was also a single k-ladder connected component (data not shown).

These results led us to the thought that, possibly, the definition of noise as bad connected nodes was wrong. We suggest that, in contrast, the noise in the network comes from nodes with very high affinity to many other nodes. Accordingly, in the clustering process we removed all nodes with a connectivity degree larger than a selected threshold. The derived graph was decomposed by the k-ladder connectivity algorithm. The results are presented in Table II. Definitely, the correctness of this suggestion should be thoroughly investigated; however, its helpfulness for clustering is apparent.

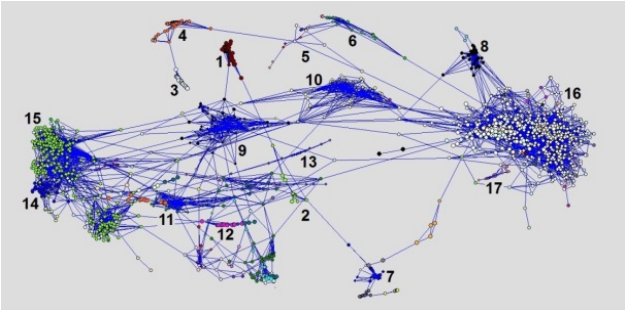


Fig. 5. 3-ladder connected components are in a good correspondence with belonging different biological functions of proteins. Only some of clusters are described:
1. Malonyl-CoA-acyl-carrier-protein; 2. MTA/SAH nucleosidase; 3. Succinyl-CoA synthetase beta; 4. Aldehyde dehydrogenase 5. Phosphomannomutase 6. Molybdenum cofactor 7. Isocitrate lyase 8. Transcriptional regulator, LysR family; 9. Transcriptional regulator, ArsR family; 10. Transcriptional regulator, AsnC family; 11. Transcriptional regulator, MarR family; 12. Transcriptional regulator, CrP family; 13. BirA bifunctional protein; 14. Transcriptional regulator, GntR family; 15. Transcriptional regulator, GntR family; 16. Transcriptional regulator, LysR family; 17. Sigma-54-dependent transcriptional regulator.

IV. Discussion

In this article we have described the development of a new network partitioning method based on the k-cycles graph connectivity approach. We formally defined a unique structure named K-Ladder connectivity. We have demonstrated that the K-Ladder-based algorithm was able to achieve the main goal, which is to successfully select the groups of functionally related proteins in the EN. In addition, the algorithm has proven to be able to process scalable networks. If needed, the algorithm may be easily implemented in a way that enables processing the whole network as a collection of segments thus, loading of the full network into the memory is not required. This is critically important for extremely large graphs such as EN.

Additional investigation is required in order to further develop this method for the use with denser networks. Although several possibilities were discussed, the practical significance of these modifications, of the basic K-Ladder algorithm, is yet uncertain. While these modifications consider much more complicated elements than a simple cycle (e.g. cliques), the advantage of the basic algorithm in speed and scalability are dramatically lost. For example, processing according to the clique percolation approach [16], [17] takes several hours for clustering, while the basic k-ladder approach takes only about a minute.

Another problem of the application of k-ladder connectivity clustering to dense networks is the possibility of high overlapping between the resulting clusters.

Although in the initial K-ladder approach there is a possibility for a node to belong to several clusters, the meaning of a connecting edge is unique. This is not the case when the unit is more complicated than cycle and connection rule requires more than just a common edge. For example, for the discussed above realization of this approach, where the connected elements of the "ladder" are cliques of a certain size, and the condition of their connection is the existence of a certain (sufficiently large) amount of common nodes, the overlapping between the selected clusters can be very high.

In some attempts to overcome these limitations, we have shown that pre-processing of the network which includes cleaning of noise by deleting poor or, oppositely, good connected nodes can potentially be fruitful. But, a biological meaning of the obtained clusters desires additional analysis.

As a final point we note that, as with all other clustering approaches, the applicability of the methods described above to other networks, different from the EN, is strongly dependent on the particular nature of the clusters, as well as on the network organization.

V. Conclusions

In this work we described a new powerful method for clustering of large Protein Evolutionary Networks. This new approach has proven to be very effective for clustering of the EN, producing a large number of meaningful small clusters. The algorithm was able to detect the clusters corresponding to different functional groups of proteins.

Several possibilities to the application of this method to other network, in particular, PPIs (protein-protein interaction networks), were also explored. The applicability of the proposed approaches to those networks requires additional analysis.

Acknowledgment

The authors are grateful to Miki Dabush and Svetlana Kolodiy for help in preliminary calculations.

References

[1] J. Wang, M. Li, Y. Deng, and Y. Pan, "Recent advances in clustering methods for protein interaction networks," *Bmc Genomics*, vol. 11, 2010.
[2] Z. M. Frenkel and E. N. Trifonov, "Evolutionary networks in the formatted protein sequence space," *Journal of Computational Biology*, vol. 14, pp. 1044-1057, Oct. 2007.
[3] Z. M. Frenkel and E. N. Trifonov, "Walking through the protein sequence space: Towards new generation of the homology modeling," *Proteins-Structure Function and Bioinformatics*, vol. 67, pp. 271-284, May 2007.
[4] Z. M. Frenkel and E. N. Trifonov, "Walking through protein sequence space," *Journal of Theoretical Biology*, vol. 244, pp. 77-80, Jan. 2007.
[5] Z. M. Frenkel, Z. M. Frenkel, E. N. Trifonov, and S. Snir, "Structural relatedness via flow networks in protein sequence space," *Journal of Theoretical Biology*, vol. 260, pp. 438-444, Oct. 2009.
[6] Y. Sobolevsky, Z. M. Frenkel, and E. N. Trifonov, "Combinations of ancestral modules in proteins," *Journal of Molecular Evolution*, Vol. 65, pp. 640-650, Dec. 2007.
[7] Z. M. Frenkel and E. N. Trifonov, "From protein sequence space to elementary protein modules," *Gene*, vol. 408, pp. 64-71, Jan. 2008.
[8] C. Walshaw and M. Cross, "Mesh partitioning: A multilevel balancing and refinement algorithm," *SIAM Journal on Scientific Computing*, vol. 22, pp. 63-80, 2000.
[9] S. Fortunato, "Community detection in graphs," *Physics Reports-Review Section of Physics Letters*, vol. 486, pp. 75-174, Feb. 2010.
[10] S. E. Schaeffer, "Graph clustering," *Computer Science Review*, vol. 1, pp. 27-64, 2007.
[11] R. Tarjan, "Depth- first search and linear graph algorithms," in *Proc. IEEE Conf Rec 1971 12th Annu Symp on Switching & Automata Theor*, pp. 114-121, 1971.
[12] J. E. Hopcroft and R. E. Tarjan, "Finding the Triconnected Components of a Graph. Technical report," CS Dept., Cornell University, Ithaca, N.Y., 1972.
[13] T. M. J. Fruchterman and E. M. Reingold, "Graph drawing by force-directed replacement," *Software-Practice and Experience*, vol. 21, pp. 1129-1164, Nov. 1991.
[14] V. Batagelj and A. Mrvar, "Pajek - Analysis and visualization of large networks," *Graph Drawing*, 2002, pp. 477-478.
[15] N. Levtov, S. Amberkar, Z. M. Frenkel, L. Kaderali, and Z. Volkovich, "Detecting Non-Uniform Clusters in Large-Scale Interaction Graphs," *Journal of Computational Biology*, vol. 21, pp. 173-183, Feb. 2014.
[16] G. Palla, A.-L. Barabasi, and T. Vicsek, "Quantifying social group evolution," *Nature*, vol. 446, pp. 664-667, Apr. 2007.
[17] J.M. Kumpula, M. Kivela, K. Kaski, and J. Saramaki, "Sequential algorithm for fast clique percolation," *Physical Review*, vol. 78, Aug. 2008.

Reshma Nibhani is a post-doctoral research scientist at Genome Diversity Center, Institute of Evolution, and University of Haifa, Israel. She was born in 1982 in Gorakhpur, U.P, and India. Dr. Nibhani received her Ph.D. degree in bioinformaticsform Center of Bioinformatics, Institute of Interdisciplinary Studies, University of Allahabad, U.P, and India. Her current research interests include sequence biology, computational biology, biophysics, protein modularity and early molecular evolution, protein structure prediction, protein folding.

Avi Soffer is a lecturer at Ort Braude College, Karmiel, Israel. He was born in 1957 in Israel. He received his B.Sc. in computer science from the Hebrew University of Jerusalem, Israel in 1986; He received his M.Sc. in software engineering from University of Maryland, College Park, USA in 1997; He received his Ph.D. in information systems engineering from the Technion – Israel Institute of

Technology in 2008. His research is focused on application of machine learning and data mining techniques in systems and software engineering, software testing, integration, and maintenance.

Ahuva Mu'alem is currently a lecturer at Ort Braude College. Before joining Ort Braude she was a post-doc fellow at the Social and Information Sciences Laboratory (SISL) at Caltech. She completed her Ph.D. at the Hebrew University of Jerusalem under the supervision of Prof. Noam Nisan. Her research is focused on computational aspects of mechanism design, in particular theoretical study and performance evaluation of auctions, scheduling and pricing policies for grid and cloud computing. She was born in Petah-Tikva, Israel in 1965.

Zeev (Vladimir) Volkovich is working as a full professor, the head of M.Sc. program in software engineering at the Software Engineering Department of ORT Braude College, Karmiel, Israel. He was born in 1953 in Tashkent (former USSR). Dr. Volkovich received his Ph.D. degree in probability theory in 1982. His current research interests include data mining, pattern recognition and clustering algorithms. He published 2 books and about 50 papers in referred international journals.

Zakharia Frenkel is working as a senior lecturer at the Software Engineering Department of ORT Braude College, Karmiel, Israel. He is also a research fellow at the Institute of Evolution, University of Haifa, Israel. He was born in 1976 in St.-Petersburg, Russia. Dr. Frenkel received his Ph. D. degree in physics and mathematics in 2003 form Physics and Mechanics Faculty, St. Petersburg State Polytechnical University. His current research interests include data mining, biophysics and bioinformatics.