Topological Fractal Dimension of Networks of Protein–Protein Interaction Networks

Georgios Kalantzis and Andrei Stoica

Systems Approaches to Biomedical Science, Doctoral Training Centre, University of Oxford

This manuscript was last compiled on November 10, 2018

Please provide an abstract of no more than 250 words in a single paragraph. Abstracts should explain to the general reader the major contributions of the article. References in the abstract must be cited in full within the abstract itself and cited in the text.

PPIN | NetworkX | BioGrid | Network Science | Centrality Measures

1. Introduction & Prerequisites

Networks are representations of real systems where individual units are modelled as nodes and interactions between these units as links. Formally speaking, this corresponds to a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, with $\mathcal{V}$ and $\mathcal{E}$ standing for the set of nodes and edges respectively. Nodes can represent anything, ranging from regions of the human brain to electrical power plants. As a result, the study of networks pervades all of science, from neurobiology to statistical physics (1).

The set of nodes $\mathcal{V}$ has usually a finite number of elements. Links can be undirected or directed, unweighted or weighted. When dealing with undirected edges, a link can be defined as a pair of nodes (u, v); in directed graphs (u, v) and (v, u) correspond to different edges. In the case of weighted networks, links are also assigned with a real number characterising the importance of the association.

There are two main ways for representing networks, namely lists and adjacency matrices. Let $\mathbf{A}$ be the adjacency matrix of graph $\mathcal{G}$ containing n nodes. Then $\mathbf{A}$ is of size $n \times n$ and element A_{ij} is 1 if nodes i, j are connected, or $A_{ij} = 0$ otherwise. In weighted networks non-zero elements of $\mathbf{A}$ are equal to the weights of the respective edges. Real complex networks, although might contain thousands of nodes, are usually sparse in regards of edges, which leads to the ability of representing by sparse matrices, special data structures efficient for computational tasks.

After defining nodes and edges, the next important term is that of node-degree. The degree k_i of node i is defined as the number of edges linked to i. In directed graphs, the degree of a node might be discriminated in in-degree and out-degree, depending on whether the edges ending to or start from i are counted. For undirected networks, the degree can be computed by

$$k_i = \sum_{j=1}^{n} A_{ij} = \mathbf{e}_i^\top (\mathbf{A} \cdot \mathbf{e}), \qquad [1]$$

where $\mathbf{e}$ stands for a column-vector full of ones and $\mathbf{e}_i$ has zeros everywhere except element i which is one. In other words, the product $\mathbf{A} \cdot \mathbf{e}$ gives the degree for every node. These are some simple indicators showing why adjacency matrices are important: they connect Network Science with Linear Algebra.

Graphs are very attractive tools to biological and medical research applications, since they can be used for the description of many mechanisms or interactions; some examples are metabolic or cell signaling networks. Another important direction are the so-called Protein-Protein Interaction Networks (which will be denoted as PPIN), which represent physical contacts between proteins within a cell. In brief, proteins are macromolecules, consisting of one or more long chains of amino acid residues, which perform a vast array of functions within organisms, including catalysing metabolic reactions, DNA replication, responses to stimuli, providing structure to cells and organisms, and transporting molecules from one location to another. Usually, the aforementioned procedures incorporate the cascade of many proteins, resulting in networks of interactions.

PPINs are characterised by three notable properties. First of all, PPINs show a small world effect meaning that there is great connectivity between proteins. More typically, the diameter (the maximum number of steps separating any two nodes) of such networks is small, regardless the number of nodes or edges. Such strong connectivity has important biological consequences, since it allows for an efficient and quick flow of signals within the network (2).

Moreover, PPINs are scale-free. This class of networks can describe a variety of complex systems where some nodes have a tremendous number edges (hubs), whereas most nodes have only a few (3). In this sense, the network appears to have no scale which provides important features. For instance, scale-free networks are very robust and stable since small perturbations have low effect. Furthermore, hubs in cancer-linked networks could be used for targeted attacks in drug discovery. Finally, another crucial characteristic of PPINs is their modularity. The transitivity or clustering coefficient of a network is a measure of nodes' tendency to cluster together. High transitivity means that the network contains communities or groups of nodes that are densely connected internally. Generally speaking, structure always affects function (1). In biological networks particularly, finding these communities is very important, because they can reflect functional modules and protein complexes.

However, PPINs are not real but correspond to actual biological networks and occur after experimental procedures. As a result, data might contain noise and some observations can be less reliable since the record of molecular interactions is occasionally incomplete or patchy.

2. Construction of PPIN and Analysis

There are various software packages or programmatic methods available to build and analyse networks. Throughout this project we work mainly with PYTHON and the NETWORKX module. Furthermore, there are a lot of sources from which you can obtain and integrate PPI data, other than creating new experimental results. Current databases are distinguished in primary or predicting, depending on whether they just provide evidence or combine other resources for prediction. For the aims of this project we will be working with data from BIOGRID, which is a primary curated biological database of protein-protein, genetic or chemical interactions as well as post-translational modifications.

A. Hands-on BIOGRID & NETWORKX . PPINs can be created from edge-lists, which are text files containing rows of the form "$\text{ID}_A - \text{ID}_B$". The latter correspond to distinctive labels of nodes which can also be used as reference to other information sources. After loading an edge-file, NETWORKX can initialise a graph with the provided edges.

In particular, we are using the dataset `BIOGRID-ORGANISM-3.5.165.tab2.zip`, offered freely in the BIOGRID repository. This dataset contains information on protein interactions for 66 organisms, stored in `Tab-2` format. Judging from the size of files, we can safely conclude that *Saccharomyces Cerevisiae, Homo Sapiens* and *Escherichia Coli K12* type W3110 have the largest numbers of protein interactions.

An Example. Let us focus on *Human Herpesvirus 6B*. After opening this file with a spreadsheet program we can easily observe that the first line (header) contains the labels of columns, followed by four lines with protein-protein interactions. What is more, there are 7 nodes in total, divided in three components with the largest containing 3 nodes. This PPIN is depicted in Figure 1 as visualized by NETWORKX .

In order to create a PPIN, what we need is a type for protein labeling. Given this dataset, there are up to three different ways, for example by using information from *Entrez* database or official symbols. We chose to work with the BIOGRID IDs since these labels are numerical and they reference directly to the online repository.

One drawback of BIOGRID is the existence of some noise in the data. By this we mean multiple copies of the same interaction, which corresponds to parallel edges, or even self-loops. In order to remove such entries we used PYTHON 's set datatype which is very efficient. Another part of pre-processing involved a re-labeling of nodes in order to construct necessary tables and arrays easier later on. To make this more helpful, we also saved a dictionary with correspondences between new and initial labels for every organism. These steps are implemented in script `ParseBioGridFiles.py` and the functions therein.

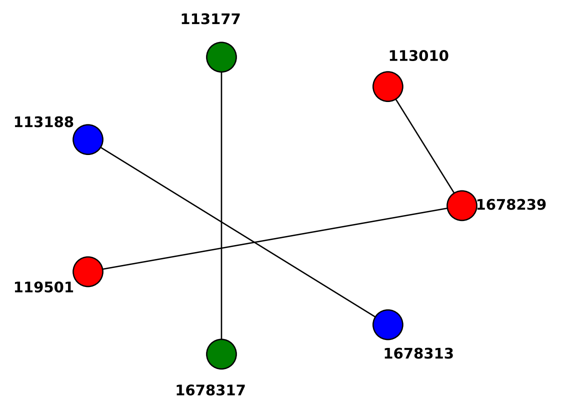

Fig. 1. A simple PPIN, the one of *Human Herpesvirus 6B*.

B. Basic Information & Illustration of PPINs. The first step of the analysis was the calculation of some fundamental properties for each PPIN, i.e. number of nodes (N), number of edges (M), average degree of nodes, number of connected components, relative size of largest connected component (LGC) and diameter. The later is computed for the LGC, for disconnected cases. All these are summarized in Table 3 (*Appendix*).

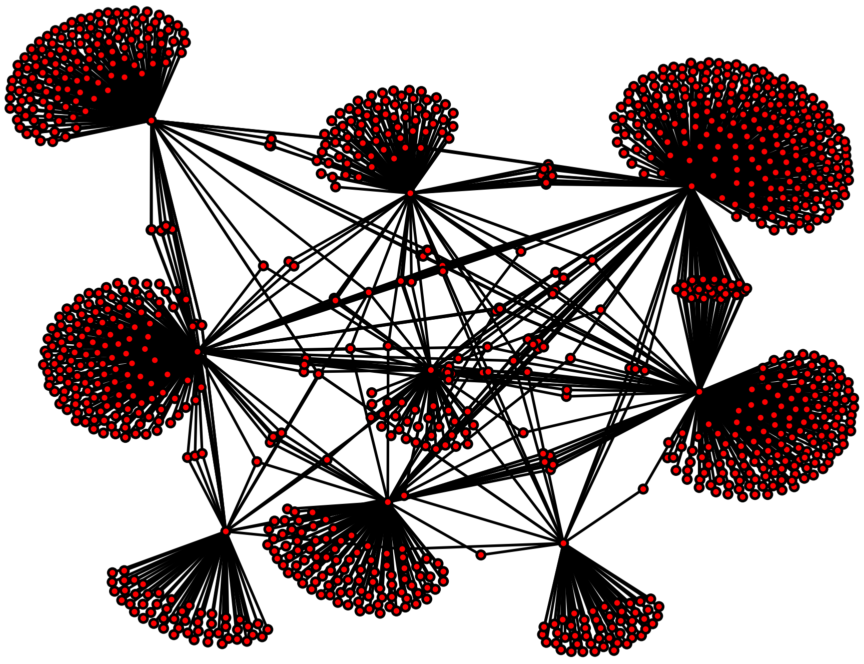

Fig. 2. PPIN of *Human Immunodeficiency Virus 1*.

Interestingly, the majority of small PPINs, i.e. those with only a few hundreds of nodes or less, show low connectivity. Two characteristic examples are *Bos taurus* (cattle) or *Danio rerio* (zebrafish) whose largest connected component contains 16% and 40%, respectively, of the total nodes. Larger networks, on the contrary, show a very different behavior regarding connectivity; almost every PPIN containing thousands of nodes has a giant component containing more than 90% of nodes. One such notable example is that of *Human Immunodeficiency Virus 1*, which is fully connected, illustrated Figure 2.

C. Centrality & Important Proteins. One of the most useful tasks in Network Science is the location of the most central or important nodes. This has always been of high importance in many scientific fields with a seminal work from Sociology aging more than six decades (4). After the enormous expansion of World Wide Web and the need of finding meaningful websites efficiently, the field of Node Ranking has gained a lot of attention the last two decades (5). To date, there is a variety of algorithms that measure centrality in order to estimate the most important ones, each with specific advantages, but also drawbacks, regarding speed, accuracy and the need of resources.

NetworkX , in particular, has a lot of built-in algorithms for node centrality, some of which we used for the analysis. First of all, `Degree Centrality` is by far an easy to compute measure and yet meaningful as it considers every direct interaction. `Katz Index` and `HITS` are algorithms that place emphasis on indirect connections. `PageRank` does also the same but in a more computationally efficient way as it incorporates many tools. The last measure we are using is `Closeness Centrality` which corresponds to the reciprocal of the average distance of a node to every other. A lot of PPINs are divided in connected components which would result in frailty when calculating closeness as some distances would be equal to infinity. To overcome this obstacle, closeness centrality is defined by an improved formula (6) as

$$C(u) = \frac{n-1}{N-1}\frac{n-1}{\sum_{v=1}^{n-1} d(v,u)}, \qquad [2]$$

where n is the size of the related component and $d(u, v)$ is shortest path (i.e. the distance) between nodes u and v. Calculating the aforementioned distances is a difficult task, however, since this is part of the next task in the project (see Section 3), we implemented a new function that calculates the closeness centralities, given the distances (check function `MyCloseness` in `PPINutils.py` script). `PageRank` and `HITS` are implemented by NetworkX in an optimal way (*scipy versions*) which, even for the largest PPINs needed only a few seconds of running time. `Katz`, on the other hand, requires the solution of a system of linear equations (or the inversion of a huge matrix otherwise), which is very time-consuming. We also tried to compute the `Betweenness Centrality` but without success, due to the high demand of processing resources.

We are now focusing on *Homo sapiens*, a PPIN with 22826 nodes, for the estimation of the most central proteins. On a typical DTC computer (3.50 GHz and 7.7 GB of RAM), `PageRank` and `Degree Centrality` took less than two seconds to run and `HITS` required approximately ten seconds. Closeness centralities were computed in almost three minutes and, finally, `Katz` took more than 45 minutes. Some of the resulting rankings seem to be consistent whereas other are dissimilar. For instance, `HITS` "agrees" with `PageRank` by $TODO$ counted by Spearmann's ρ ranking coefficient.

In order to select the most important nodes we do the following procedure: we select the top-15 ranked nodes by each of the five algorithms, create a list with all these 75 elements and count the multiple occurrences. The most frequent proteins are reported in Table 1 accompanied by the corresponding gene and protein function.

Table 1. Top-5 Central Proteins in *Homo Sapiens*

Node Label	Protein	Gene	Function
114030	Cullin-3	CUL3	This protein plays a critical role in the polyubiquitination and subsequent degradation of specific protein substrates
113164	HMG20	UBC	It plays a key role in maintaining cellular ubiquitin levels under stress. Defects could lead to embryonic lethality.
108309	HuR	ELAVL1	RNA-binding protein that binds to the 3'-UTR region of mRNAs and increases their stability
113348	Exportin-1	XPO1	eukaryotic protein that mediates the nuclear export of proteins, rRNA, snRNA, and some mRNA.
113010	TP53	TP53	tumor suppressor protein containing transcriptional activation, DNA binding, and oligomerization domains.

D. Degree Distributions. We move to another interesting aspect of complex networks which concerns the distribution of degrees. Although many of the constructed PPINs are interesting, for the sake of clarity we focus mainly on those by *Homo sapiens* and HIV. We present such information in Figures 3-4. In general, as was also mentioned in Section 1, most of the PPINs are scale-free since there are hubs that are connected with a large proportion of nodes and, on the same time, a lot of nodes have only a few edges. This behavior, which is explained by the fact that degree distributions of such networks usually follow a power-law (3), can be observed by the histograms (left part of the following figures).

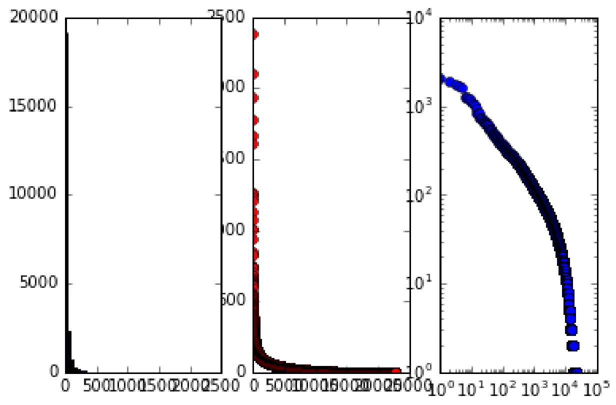

Fig. 3. Degree distribution in *Homo sapiens*

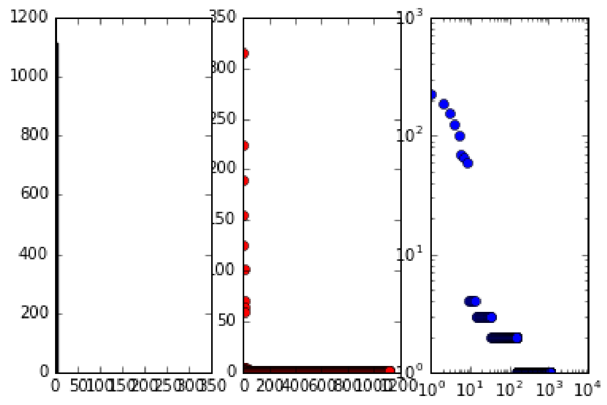

Fig. 4. Degree distribution in *HIV-1*

E. Computational Complexity. We are now investigating the time complexity for PPIN construction. There might be a variety of different ways to study complexity but we measure the total time needed for loading an edge-file with Python and adding the set of edges in a NetworkX graph. We assume that intermediate actions are the same for every organism and try to find a more abstract relationship between time (T) and size of graphs ($N or M$).

After measuring the time complexity for each of the 66 cases, we construct two plots in a logarithmic scale, as presented in Figures 5-6. We also plot a corresponding line after fitting a linear regression model. Despite the existence of some oscillations in small graphs, there seems to be a strong linear relationship between T and M and a less accurate linearity between T and N. Speeding-up the construction of PPINs would require a deeper understanding of

PYTHON 's commands for file I/O. For instance, working with `Pandas` module instead of `csv.reader` gives a high advantage when constructing a graph from the scratch. On the other hand, we noticed that `csv.reader` was the only feasible way to read large files with node-distances.

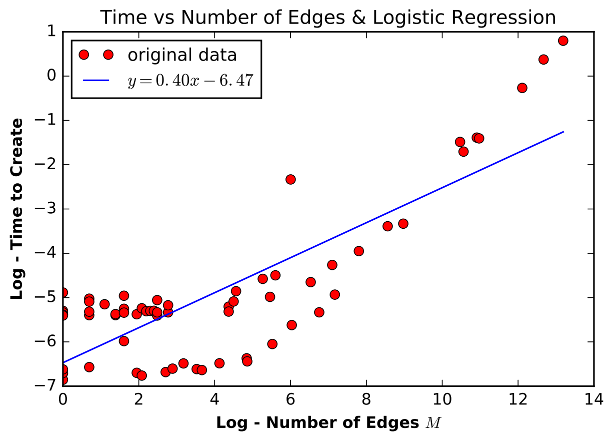

Fig. 5. Time vs Number of Edges

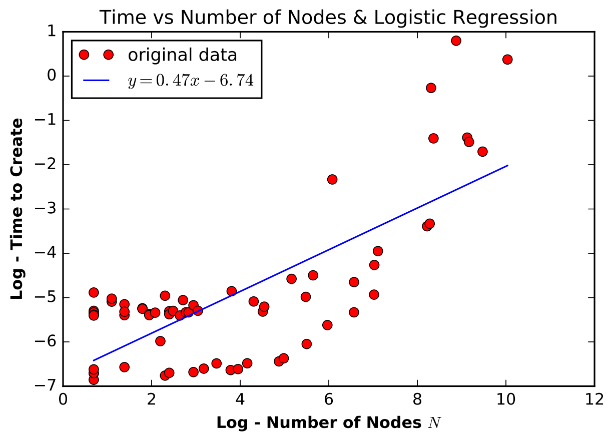

Fig. 6. Time vs Number of Nodes

3. Computing the Topological Fractal Dimension

A. The Box Counting Method. Before we can introduce the concept of topological fractal dimension of a network, a preamble through the idea of box covering (originally due to Hausdorff) is necessary. Given a network G and a box size l_B, a box covering of the network consists of an ensemble of disjoint boxes that together cover every node, with the property that the distance between any two nodes in each box is less than the given l_B value. For any l_B we define N_B as the minimum number of boxes that are required for such a covering. It is thus clear that for $l_B = 1$ we have that $N_B = N$, which is defined as the number of nodes. As we increase l_B, N_B will correspondingly decrease, until a value of 1 is obtained if the network is connected (or more generally, the minimal attainable N will equal the number of connected components in the graph). We define the l_B for which N_B reaches its minimum as l_B^{max}

To identify the correct N_B for a given l_B in a reasonable amount of time, box covering algorithms have been developed. The problem of box covering is known to be NP-hard - that is, no algorithm will produce an exact solution in a reasonable amount of time. Therefore, existing algorithms only aim to optimally approximate N_B, rather than calculate it exactly.

For any given network, the N_B and the l_B values have been shown to satisfy a relation of the form:

$$N_B \sim l_B^{-d_B} \qquad [3]$$

The exponent d_B is defined as the topological fractal dimension of the network. To determine it, first we employ a box covering algorithm to calculate the N_B. The method we chose was a greedy algorithm, as described in (7), with the following steps:

1. Initialize the network, numbering the nodes from 1 to N
2. For all l_B values, assign a box number of 0 for the first node.
3. For l_B values from 1 to l_B^{max}, repeat the following:

   - For node i (starting at node 2) mark all box numbers where the distance between i and j is greater or equal than l_B as unavailable
   - Select one box number from the remaining available ones for node i
   - Increase i till it reaches N
   - Increase l_b till it reaches l_b^{max}

B. Implementation. This method necessitates information about the distances between every pair of nodes in the graph. This can be calculated as the algorithm is run (conveniently using the shortest path function from NetworkX) or alternatively pre-calculated before for every pair and stored in a matrix that will be read when the program that calculates the TFD starts. Due to speed constraints, we found the second approach to be preferable; as such, we wrote a C script to calculate the distance matrices for all 66 different PPINs and we stored each one of then in a different file. Originally, the algorithm that we chose to calculate all the distances was Floyd-Warshall, due to simplicity of implementation. However, the speed of this dynamic programming algorithm proved insufficient for our needs, and we therefore switched to a breadth-first search (BFS) implementation, again done in C. The speed of computations was now satisfactory, and we managed to compute all the distances in our PPINs and then integrate them into the Python program in order to calculate the number of boxes.

C. Testing the Method. The same Python program was then used to calculate the TFD. To that direction, we rewrite Eq. 3 using a constant of proportionality C, then we take natural logarithms and rewrite everything in matrix form, forming a system of linear equations, as follows:

$$\begin{bmatrix} \log N_1 \\ \log N_2 \\ \vdots \\ \log N_{l_B^{max}} \end{bmatrix} = \begin{bmatrix} 1 & -\log l_1 \\ 1 & -\log l_2 \\ \vdots & \vdots \\ 1 & -\log l_B^{max} \end{bmatrix} \begin{bmatrix} \log C & d \end{bmatrix} \qquad [4]$$

We solved for C and d using the linear least squares method from Python 's *scipy* library. Before proceeding with the calculation on PPINs, we tested this greedy algorithm on synthetic networks. Three types of networks were considered: simple path graphs, lattice graphs and Erdos-Renyi graphs. In the path graph, where all the edges are between nodes with consecutive indices, we found that the value of the TFD seems to stabilize around 0.85 as we increase the number of nodes N. The theoretical predicted value was approximately 1, and the differences between it and the calculated value are likely to be due to the simplifying assumptions employed in order to find the theoretical value (detailed upon in the next section).

Regarding the lattice graph, where nodes form a grid, the expected value was 2. Our algorithm would obtain values as small as 1.2 for some of the smallest lattice graphs (e.g.$n = 3$), but it would continuously increase and eventually reach a plateau around 2.35 (obtained for very large lattice graphs, with $n > 230000$). The discrepancy between 2.35 and 2 is partly due to the fact that on lattice graphs, the greedy algorithm is not guaranteed to find the optimal solution, so it overestimates N_B for certain values of l_B.

Finally, our testing on Erdos-Renyi graphs (random graphs for which edge has a probability $p \in [0, 1]$ to be included in the graph

D. Analytical Expressions of the TFD of Synthetical Networks. Certain simple synthetical networks behave in predictable ways, rendering calculations of TFD feasible. The most immediate example is the complete graph, which has an edge between every pair of nodes. Thus, a box covering of size 2 will require just 1 box. Corroborating this with the fact that a box covering of size 1 requires n boxes, we observe that $C = n$ and $d = \log_2^n$ fit our equation $N_B \approx C l_B^{-d_B}$

Another such example is the path graph. By construction, any distance between two nodes in this graph equals the absolute value of the difference of the two corresponding indices. Therefore, determining the optimal box covering is trivial: a box of size l can contain at most l nodes. So the total number of required boxes will be $\left\lceil \frac{n}{l_B} \right\rceil$ (here $\lceil \quad \rceil$ denotes the ceil function, defined as the smallest integer larger or equal than n).

Plugging back into our original equation, we arrive at this:

$$\left\lceil \frac{n}{l_B} \right\rceil \sim l_B^{-d_B} \qquad [5]$$

Asymptotically $\left\lceil \frac{n}{l_B} \right\rceil \sim \frac{n}{l_B}$, so $\frac{n}{l_B} \approx C l_B^{-d_B}$, where C is a proportionality constant. We can then observe that equality is attained when both $C = 1$ and $d_B = 1$ - thus, benefiting from the approximation of the ceiling function of $\frac{n}{l}$, we can use $d_B = 1$ as an adequate approximation of the topological fractal dimension for path graphs. Notably, the greedy algorithm is exact for path graphs; it will always provide the correct number of boxes.

4. TFD of PPIN & Conclusions

We are now ready to comment on the final part of our project, which is the TFD calculation for various PPINs. After implementing the box covering algorithm, we tested it in two versions of PPINs for each of the 66 organisms, on the complete graph with isolated nodes or disconnected components, as well as on the largest connected component (LCC). The results could be grouped in the following three cases. Firstly, the fully connected networks which showed no difference, those which had disconnected components but TFD was on similar values and, thirdly, some organism which resulted in significantly different TFD between the whole graph and the largest connected component.

Table 2 summarizes the TFD of several PPIN, for both scenarios. $Anopheles gambiae PEST$ is an organism, among other, whose trivial PPIN – two nodes with one edge – is depicted in a TFD of 1. *HIV-1*, as stated earlier, has a fully connected PPIN and this is in accordance with the equality of the two TFDs.

Table 2. TFD calculation of PPINs on full graph or LCC

	Organism	TFD-full	TFD-LCC
1	$Anopheles\ gambiae\ PEST$	1.0	1.0
2	$Arabidopsis\ thaliana\ Columbia$	1.936	3.68
3	$Dictyostelium\ discoideum\ AX4$	1.171	1.342
4	$Escherichia\ coli\ K12\ W3110$	4.913	4.913
5	$Hepatitus\ C\ Virus$	3.399	3.948
6	$Homo\ sapiens$	3.469	4.725
7	$Human\ Herpesvirus\ 1$	2.466	2.466
8	$Human\ Herpesvirus\ 6B$	0.775	0.585
9	$Human\ Immunodeficiency\ Virus\ 1$	4.395	4.395
10	$Saccharomyces\ cerevisiae\ S288c$	4.822	4.822

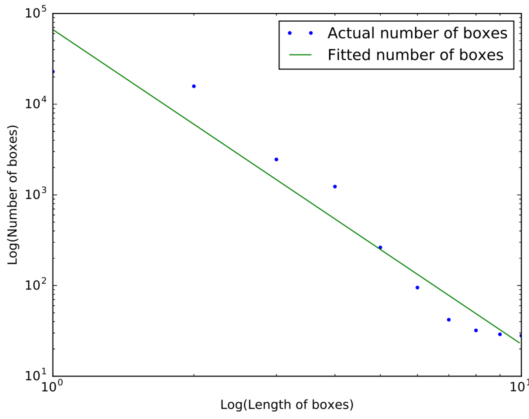

Fig. 7. Data fitting on *Homo sapiens* complete PPIN

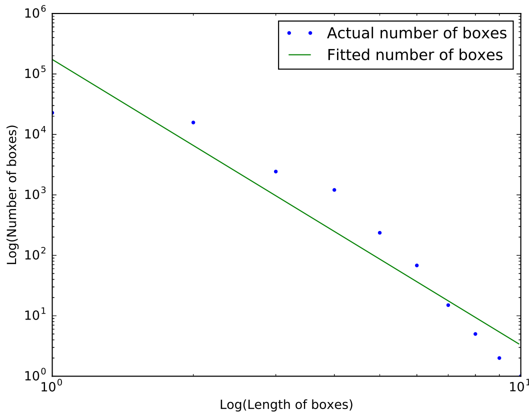

Fig. 8. Data fitting on the largest connected component of Homo sapiens PPIN

The diagrams of Figures 7-8 present the number of boxes required for graph covering as a function of the box-side length, in a logarithmic scale. The estimated pairs are depicted as blue dots whereas green lines correspond to the linear-fit approximation.

Code and Scripts. For the sake of efficiency and productivity we used a GitHub repository [*] with all our scripts and functions. After providing the initial dataset, it should work as a pipeline following the directions in comments or descriptions.

References

1. Strogatz SH (2001) Exploring complex networks. *nature* 410(6825):268.
2. Institute EB (year?) Network analysis of protein interaction data: an introduction (https://www.ebi.ac.uk/training/online/course/network-analysis-protein-interaction-data-introduction). Accessed: 2018-10-31.
3. Barabási AL, Bonabeau E (2003) Scale-free networks. *Scientific american* 288(5):60–69.
4. Katz L (1953) A new status index derived from sociometric analysis. *Psychometrika* 18(1):39–43.
5. Langville AN, Meyer CD (2011) *Google's PageRank and beyond: The science of search engine rankings.* (Princeton University Press).
6. Wasserman S, Faust K (1994) *Social network analysis: Methods and applications.* (Cambridge university press) Vol. 8.
7. Song C, Gallos L, Havlin S, Makse H (2007) How to calculate the fractal dimension of a complex network: the box covering algorithm. *Journal of Statistical Mechanics: Theory and Experiment* 2007(03):P03006.

[*] github.com/giorkala/Protein-Protein-Interaction-Networks-Project/tree/master/PPIN_Construction

Table 3. ***Appendix*** **- Basic Information about PPINs from** `BioGrid 3.5.165 dataset`

	Organism	# Nodes	# Edges	Avg Degree	ConComps	Largest	Diam
1	*Anopheles gambiae PEST*	2	1	1.00	1	1.000	1
2	*Apis mellifera*	2	1	1.00	1	1.000	1
3	*Arabidopsis thaliana Columbia*	9570	35242	7.37	78	0.981	12
4	*Bacillus subtilis* 168	2	1	1.00	1	1.000	1
5	*Bos taurus*	437	405	1.85	73	0.162	15
6	*Caenorhabditis elegans*	3938	7885	4.00	77	0.953	13
7	*Candida albicans SC*5314	713	860	2.41	30	0.893	12
8	*Canis familiaris*	52	34	1.31	20	0.135	4
9	*Cavia porcellus*	9	5	1.11	4	0.333	2
10	*Chlamydomonas reinhardtii*	19	15	1.58	4	0.632	2
11	*Chlorocebus sabaeus*	11	7	1.27	4	0.273	2
12	*Cricetulus griseus*	32	24	1.50	8	0.500	3
13	*Danio rerio*	245	250	2.04	37	0.404	8
14	*Dictyostelium discoideum AX*4	24	18	1.50	6	0.208	2
15	*Drosophila melanogaster*	9191	54806	11.93	37	0.992	9
16	*Emericella nidulans FGSC A*4	64	62	1.94	6	0.703	2
17	*Equus caballus*	4	2	1.00	2	0.500	1
18	*Escherichia coli K*12	2	1	1.00	1	1.000	1
19	*Escherichia coli K*12 *MC*4100 *BW*2952	10	8	1.60	2	0.800	6
20	*Escherichia coli K*12 *MG*1655	146	127	1.74	21	0.623	3
21	*Escherichia coli K*12 *W*3110	4063	181620	89.40	1	1.000	5
22	*Gallus gallus*	391	417	2.13	42	0.588	9
23	*Glycine max*	44	39	1.77	7	0.318	2
24	*Hepatitus C Virus*	131	129	1.97	2	0.985	2
25	*Homo sapiens*	22826	318912	27.94	14	0.999	9
26	*Human Herpesvirus* 1	174	194	2.23	1	1.000	8
27	*Human Herpesvirus* 2	7	4	1.14	3	0.429	2
28	*Human Herpesvirus* 3	4	2	1.00	2	0.500	1
29	*Human Herpesvirus* 4	240	235	1.96	7	0.771	8
30	*Human Herpesvirus* 5	91	79	1.74	12	0.385	4
31	*Human Herpesvirus* 6*A*	11	7	1.27	4	0.364	2
32	*Human Herpesvirus* 6*B*	7	4	1.14	3	0.429	2
33	*Macaca mulatta*	15	12	1.60	3	0.733	2
34	*Human Herpesvirus* 7	2	1	1.00	1	1.000	1
35	*Meleagris gallopavo*	2	1	1.00	1	1.000	1
36	*Human Immunodeficiency Virus* 1	1121	1299	2.32	1	1.000	5
37	*Human Immunodeficiency Virus* 2	16	12	1.50	4	0.500	4
38	*Human papillomavirus* 16	14	12	1.71	2	0.857	4
39	*Mus musculus*	13003	38624	5.94	89	0.984	15
40	*Neurospora crassa OR*74*A*	12	10	1.67	2	0.667	2
41	*Mycobacterium tuberculosis H*37*Rv*	11	9	1.64	2	0.818	2
42	*Nicotiana tomentosiformis*	2	1	1.00	1	1.000	1
43	*Oryctolagus cuniculus*	283	271	1.92	33	0.502	9
44	*Oryza sativa Japonica*	74	90	2.43	18	0.351	3
45	*Pan troglodytes*	10	5	1.00	5	0.200	1
46	*Ovis aries*	2	1	1.00	1	1.000	1
47	*Pediculus humanus*	2	1	1.00	1	1.000	1
48	*Plasmodium falciparum* 3*D*7	1224	2445	4.00	23	0.963	10
49	*Rattus norvegicus*	3713	5227	2.82	118	0.918	14
50	*Ricinus communis*	3	2	1.33	1	1.000	2
51	*Saccharomyces cerevisiae S*288*c*	7158	534073	149.22	1	1.000	6
52	*Schizosaccharomyces pombe* 972*h*	4292	58217	27.13	7	0.997	8
53	*Selaginella moellendorffii*	6	8	2.67	1	1.000	2
54	*Simian Immunodeficiency Virus*	19	16	1.68	4	0.421	3
55	*Simian Virus* 40	6	5	1.67	1	1.000	2
56	*Solanum lycopersicum*	45	96	4.27	7	0.467	4
57	*Solanum tuberosum*	4	2	1.00	2	0.500	1
58	*Sus scrofa*	94	79	1.68	23	0.234	5
59	*Strongylocentrotus purpuratus*	17	16	1.88	1	1.000	2
60	*Vaccinia Virus*	8	5	1.25	3	0.375	2
61	*Tobacco Mosaic Virus*	3	2	1.33	1	1.000	2
62	*Ustilago maydis* 521	4	3	1.50	1	1.000	2
63	*Xenopus laevis*	1128	1212	2.15	61	0.850	15
64	*Vitis vinifera*	2	1	1.00	1	1.000	1
65	*Zea mays*	21	11	1.05	10	0.143	2
66	*Human Herpesvirus* 8	714	687	1.92	45	0.529	12