Topological Fractal Dimension of Protein-Protein Interaction Networks

Mahan Ghafari, Rocco Meli, and Florian Störtz

Interdisciplinary Bioscience DTP, Rex Richards Building, University of Oxford, Oxford OX1 3RQ

November 11, 2018

Protein-protein interaction networks are attracting increasingly more attention in recent years thanks to the rapid development of new methods to study protein structures, functions, and interactions. These networks provide a level of abstraction that enables us to understand some of the global features of protein interactions in cells. A large number of them exhibit scale-free and scale-invariant behavior. Here, we measure the fractal dimension of some real cellular networks using three different box counting methods. We also put forward a new method for measuring fractal dimension based on the box burning method. We find that a power-law relation between the number of boxes needed to cover the networks and the size of the boxes does indeed exist, and subsequently compare the performance and limitations of each method.

Topological Fractal Dimension | Protein-Protein Interaction Networks

Protein-protein interactions (PPIs) are the basis of many important cellular processes such as signal transduction, molecular transport and various metabolism pathways. Aberrant PPIs are responsible for many aggregation-related diseases, such as Alzheimer's disease (1), and may also lead to cancer (2). Understanding PPIs is also essential in drug development, since drugs can affect them by manipulating active sites of proteins. Therefore, they have been studied extensively in the area of bioscience and medical research (3–5). Proteins interact with each other on a physical level due to the combined effects of hydrophobic bonding, salt bridges and van der Waals forces at specific binding domains. Their interaction has several biochemical effects (6, 7), which can be used for detection: they can alter the kinetic properties of enzymes affecting their binding to substrates, create new binding sites, inactivate a protein or result in a new function that neither interaction partner had before. Early methods like yeast two-hybrid screening (8) relied on the latter effect, i.e. the altering of gene expression in a reporter gene mediated by the interaction of two specific proteins *in vivo*. The method had first been established for *Saccharomyces cerevisiae*, which therefore offers the most extensive data set to date (9). The development of large-scale PPI screening techniques, especially high-throughput affinity purification combined with mass spectrometry (10), has caused an explosion in the amount of PPI data and the construction of ever more complex and complete interactomes, i.e. the totality of PPIs that happen in cells and organisms. This method involves the selective purification of a tagged protein and its interaction partners, and subsequent analysis via mass spectrometry. Our current knowledge of interactomes is both incomplete and noisy, as all the current PPI detection methods have their limitations as to how many truly physiological interactions they can detect. Experimental results have since been catalogued and curated for several online databases (9, 11) in the form of single undirected interaction data points.

Protein-protein interaction networks (PPINs) are mathematical representations of the physical contacts between proteins in the cell. These contacts are specific binding regions in the protein that serve a specific function. Knowledge of PPIs enables us to assign putative roles to uncharacterized proteins, add fine-grained detail about the steps within a signalling pathway, and characterize the relationships between proteins that form multi-molecular complexes such as the proteasome (12). Understanding the structure of these interaction networks only provides one dimension of the biochemical machinery controlling cell behavior. Apart from cellular networks, one might also study metabolic networks (13), gene expression networks (14), and domain interaction networks (15).

Due to the great level of connectivity between proteins, PPINs demonstrate small-world effects. This means that, no matter how big the PPIN is, the maximum number of steps separating any two protein interactions (i.e. nodes in the network) is small. Several studies have shown that real complex networks, ranging from the world wide web to protein-protein interaction networks, have self-repeating patterns over a wide range of length scales (16–18).

In this work, we first import protein interactions from the BioGRID online database (https://www.thebiogrid.org/, build 3.5.165 of 23rd Sep 2018) (9), into the Python network manipulation package `networkx`. To unfold the self-similar properties of PPINs, we then calculate their topological fractal dimension using several possible box covering algorithms (16–19). We also put forward a new method for calculating the topological fractal dimension using the 'burning method'. We then compare these methods against some generic network structures as well as cellular networks of *Escherichia coli* and *Arabidopsis thaliana*. Finally, we discuss the results and compare the limitations and efficiency of each method.

Data Import and Analysis

The Python library `pandas` was used to import BioGRID files into a `networkx` graph. The latter automatically cleaned the data. An execution time analysis of this process is depicted in Figure 1, showing a power law dependence of the execution time on the number of edges M across several orders of magnitude.

In order to quantify the importance of a node, we calculated three centrality measures for each node (see Figures 2a and 3). Betweenness quantifies how often a node is part of the shortest path between two other nodes, eigenvector centrality is the

Table 1. Definitions

Term	Definition
Node (or vertex)	A protein in a network
Edge (or link)	Functional (or physical) interaction between proteins
Degree	The number of incoming/outgoing edges
Hub	A node with a high degree
Shortest path length	Smallest number of steps (edges) to connect a pair of nodes
Diameter	Maximum shortest path length in the network
Local diameter nodes	All nodes separated from a local starting node by equal length

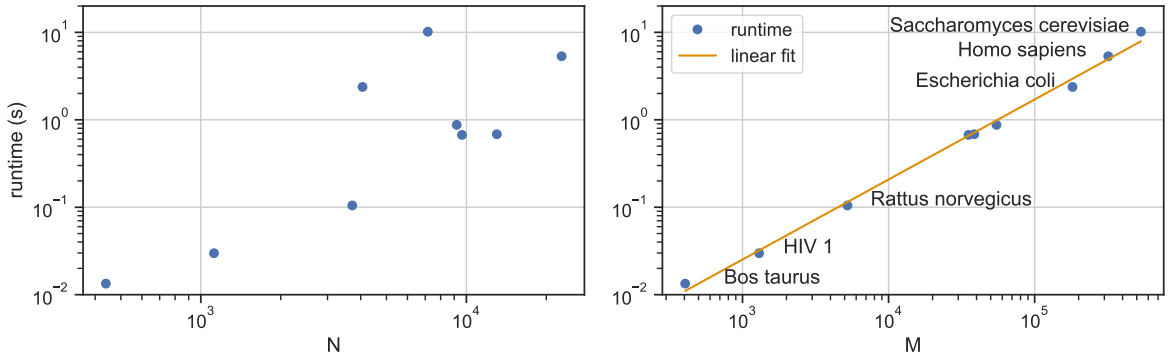


Fig. 1. Runtime analysis of building the graphs from the BioGRID files for several organisms using the Python library `networkx`. This involves deletion of selfloops and duplicates in the graph. We observe a power law dependence of the runtime on the number of edges M, whereas the dependence on the number of nodes N is not as uniform.

sum over the eigenvector centrality of neighboring nodes, and the PageRank value, which in an undirected graph as ours is proportional to the degree of a node (20).

Figure 2a shows the Ubiquitin-C (Ubc) protein to be the one with the highest centrality in *Rattus norvegicus* by our chosen measures. The reason for this is that it is frequently found in covalent bonds with other proteins or even itself (polyubiquitin chains) and, depending on its binding partner, can assist in DNA repair, cell-cycle regulation, endocytosis and catabolic processes in muscle cells (22).

Figure 3 shows the same centrality measures for the highest scoring human proteins. We observe that the TRIM25 protein, whose RING-type region is involved in viral immune defense (23), has the highest overall centrality. NTRK1, which is a nerve growth factor receptor, has the highest eigenvector centrality and has a pro-survival signalling effect in neurons (24). Together with CUL3, it has been shown to act as a hub protein (25).

Box-covering Algorithm

In order to compute the fractal dimension d of a network, we use a box-covering algorithm. The idea behind a box-covering algorithm is to find the minimum number of boxes of a given size needed to cover all the nodes of the complex network. The fractal dimension of a complex networks relates the minimum number of boxes N_B and the size of the box l_B as

$$N_B \sim l_B^{-d}. \quad [1]$$

Therefore, in order to compute the fractal dimension of a network we have to compute N_B for different values of l_B and perform a power-law fit in order to obtain d.

In this work, we implemented three different methods to compute the fractal dimension of a network: the burning algorithm, the greedy coloring algorithm and the fuzzy algorithm.

Effect of Selecting a Starting Node on the Total Number of Boxes

To obtain an estimate for the fractal dimension, we have to ensure that the total number of boxes (or colors) is minimal. One way to test this is to see how N_B changes with the starting node of a compact box of length l_B. Figure 4 shows why picking a node with the least number of neighbors can provide a smaller number of boxes than randomly selecting nodes to form compact boxes. We put this into test by comparing the two strategies in estimating the fractal dimension of a lattice (see Figure 5). We find that the fractal dimension of a random model $d \approx 1.60$ is less accurate than selecting nodes with minimum adjacent neighbors, for which we obtain $d \approx 1.75$. Additionally, the distribution of box counts tends to be much more spread around the mean for the former method.

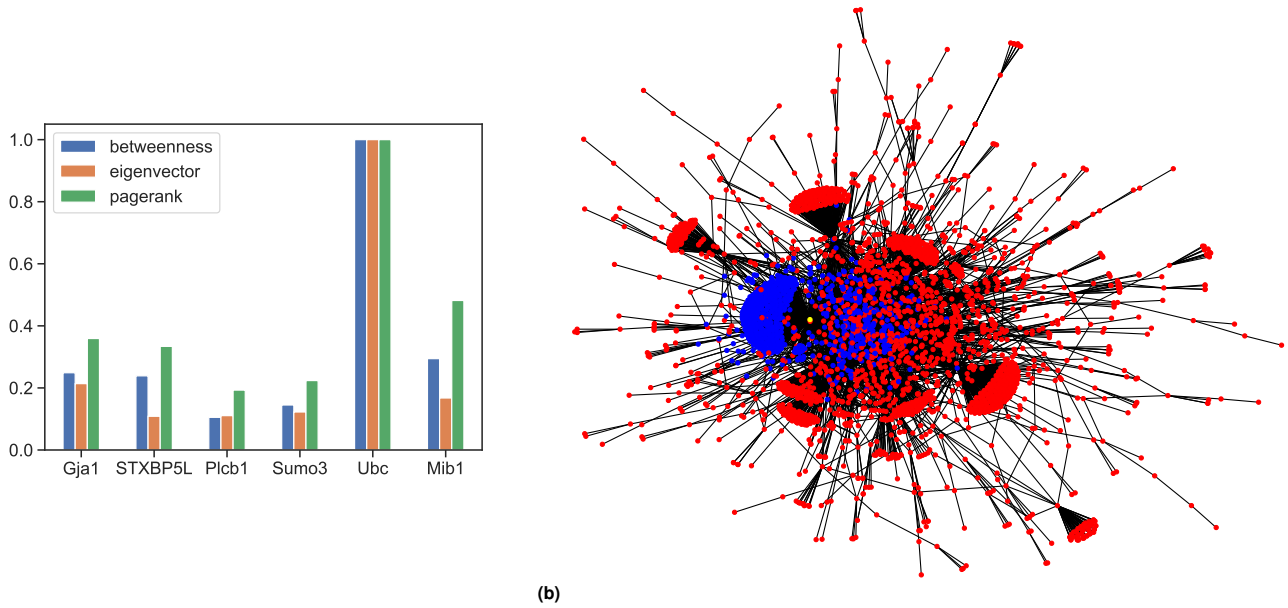


Fig. 2. Network analysis of the protein interactions in *Rattus norvegicus*, the biggest part of which were obtained experimentally in Reference (21). **a)** Centrality measures for the highest-scoring proteins, where the values have been normalized such that the maximum score for each measure equals unity. For the PageRank centrality, we chose a damping factor of $\alpha = 0.85$. The Ubiquitin-C (Ubc) protein is the most central for all three measures. **b)** Visualization of the largest connected component (3254 proteins with 5013 interactions). The central part is densely interconnected, with several distinct bunching points, and shows outgoing interaction chains. The yellow dot in the center-left region represents the Ubc protein which is the most central in the network; blue dots represent proteins which immediately interact with Ubc.

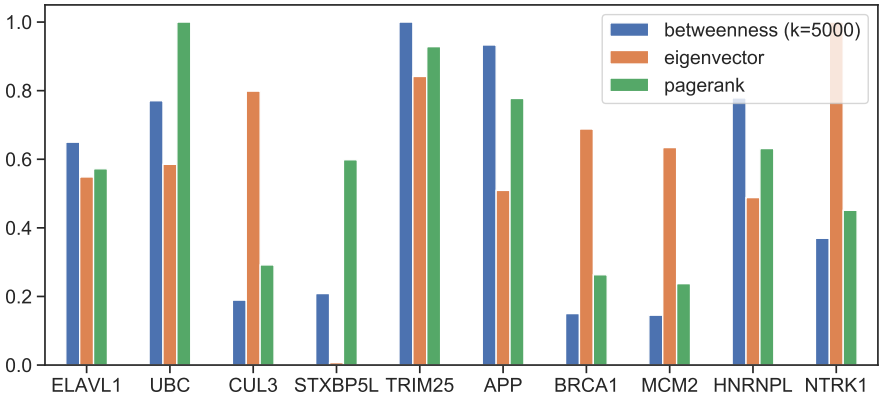


Fig. 3. Centrality measures for *H. sapiens*. As it is computationally costly, we only used $k = 5000$ randomly sampled nodes when evaluating the betweenness of each node.

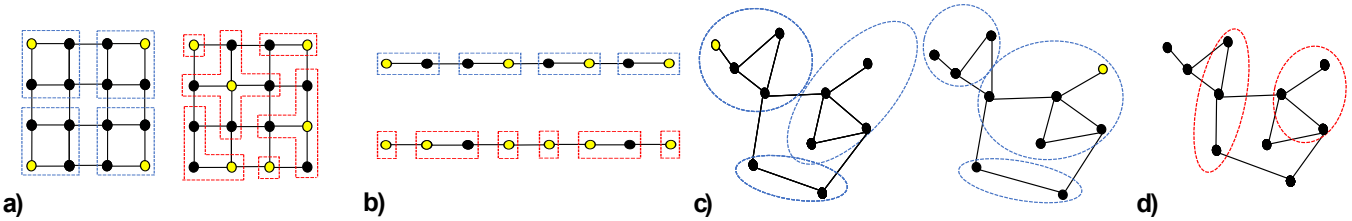


Fig. 4. Counting the number of boxes in **a)** a lattice and **b)** a path network. Yellow points show the starting nodes of a compact box (dashed lines). If the starting point is from nodes with the least number of neighbors, the total number of compact boxes (blue lines) is minimum. On the other hand, if the starting nodes are selected randomly, the total number of boxes (red lines) could be higher. Panel **c)** shows that the total number of boxes tends to stay the same (minimum) as we randomly select different starting points with the least number of neighbors. Panel **d)** shows an example of making non-compact boxes (red lines).

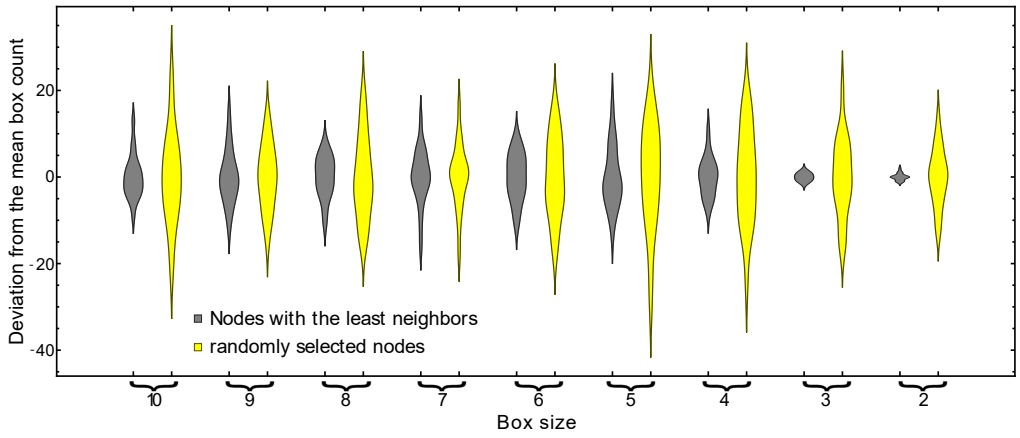


Fig. 5. The effect of choosing the initial node of a compact box on the total box count. For a 50×50 lattice, we count the minimum number of boxes by randomly selecting nodes in the lattice (gray) and randomly selecting nodes with the least number of neighbors (yellow) to form the minimum set of compact boxes. We run 20 independent simulations and vary the box size from 10 to 2 to get a distribution over the total number of boxes using each approach (for a box size of length 1, the total box count will be equal to the total number of nodes in the lattice, i.e. 2500). We find that the variation in the total number of boxes is much higher for randomly selected nodes.

Finite-Size Effects

In order to test our algorithms, we apply them to two simple networks: a linear and a lattice network (see Figure 6). For the linear network we expect an integer dimension $d_{\text{expected}} = 1$ while for the lattice network we expect an integer dimension $d_{\text{expected}} = 2$. When we apply the algorithms to these networks, we found that the fractal dimensions are smaller than the expected dimensions.

To investigate this behaviour we applied the fuzzy algorithm to networks with an increasing number of nodes, the results of which are reported in Figure 7a. We see that when increasing the linear number of nodes (the number of nodes for the linear network and the square root of the number of nodes for the lattice network), the normalized fractal dimension d/d_{expected} approaches the correct limit. However the convergence rate decreases, as can be inferred from Figure 7a.

Finally, to better understand this phenomenon and see whether it is related to the finite size of the networks, we applied periodic boundary conditions (PBCs). These are commonly employed to simulate infinite systems (26) and therefore we expect such boundary conditions to lower the extent of finite-size effects. As we can see from Figure 7a this is indeed the case: the use of periodic boundary conditions shifts the fractal dimension towards the expected limit. Surprisingly, the use of periodic boundary conditions does not seem to increase the convergence rate with respect to the linear number of nodes.

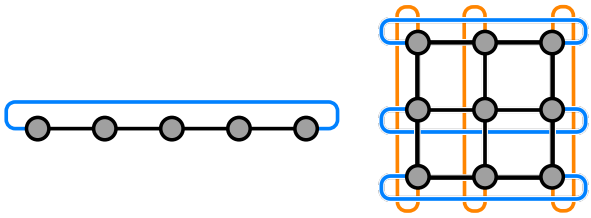


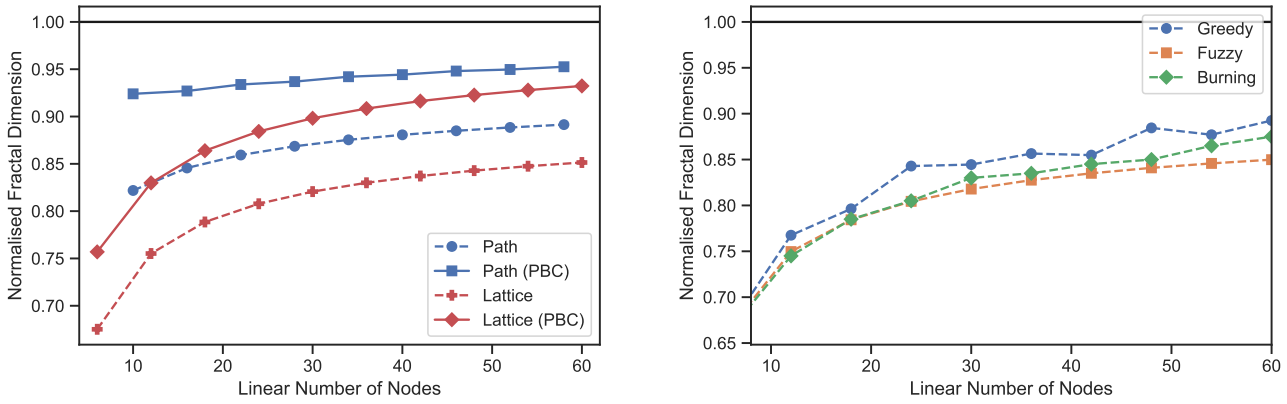
Fig. 6. (Left) Linear (or path) network with $N = 5$ nodes and **(right)** Lattice network with $N = 9$ nodes. Black lines represent the edges without periodic boundary conditions, i.e. the path network is just a 1D line and the lattice network is a 2D grid. Colored lines represent the edges added to obtain periodic boundary conditions: topologically, the path network becomes a ring while the lattice network becomes a torus.

Comparison of Different Methods to Compute the Fractal Dimension of a Network

We implemented three different algorithms: the burning method (16–18), the greedy coloring algorithm (18) and the fuzzy algorithm (19), comparing their relative performance.

Figure 7b shows the normalized fractal dimension $d/2$ for a lattice graph (without periodic boundary conditions) as a function of the linear number of nodes in the lattice for the three different algorithms. We see that for the lattice graph, they give a comparable fractal dimension and have a similar trend. This result shows that for networks with a simple structure the three algorithms produce identical results.

The greedy algorithm was run only once and therefore we could not perform averaging. This accounts for the fluctuations observed for the greedy algorithm in Figure 7b, since the algorithm is not deterministic (18) and an average over a large number of runs is expected to yield more accurate results.



(a) (b)

Fig. 7. (a) Finite-size effects for path and lattice graphs. The fractal dimension of both graphs slowly converges to the exact limit $d/d_{\text{expected}} = 1$ with an increasing number of nodes. The use of periodic boundary conditions improves the fractal dimension, providing a vertical shift towards the exact limit, but does not appear to improve the convergence rate. **(b)** Normalized fractal dimension $d/2$ for a lattice network (without periodic boundary conditions) as a function of the linear number of nodes in the lattice for the three different algorithms (burning, fuzzy and greedy).

Table 2. Fractal Dimension d for different PPINs with comparison values. Only the values marked with $\star$ are known to stem from BioGRID data.

Organism	N	M	Fuzzy	Burning	Greedy	Song *et al.* (16)	Wang *et al.* (27)	Wei *et al.* (28)	Jin *et al.* (29)
Human herpesvirus 8	714	687	2.30	2.20	2.32				
Caenorhabditis elegans	3,953	7,885	3.86	1.39	2.31		3.48 ± 0.24 $\star$	1.704	1.6±0.1
Arabidopsis thaliana	9,626	35,242	4.16	1.83	2.61		2.26 ± 0.06 $\star$		1.5±0.1
Escherichia coli K-12	4,063	181,620	5.10	3.44	4.51	2.3	2.37 ± 0.11	3.351	3.6±0.3

Fractal Dimension d of Real Protein-Protein Interaction Networks

After testing our algorithms and their properties, we applied them to real protein-protein interaction data obtained from the BioGRID database. Table 2 reports the fractal dimensions we obtained using three different algorithms: a novel box burning (based on Refs. (16–18)), a greedy coloring (18) and the fuzzy method (19).

From Table 2, we see that the fractal dimensions obtained with the three different algorithms do not always match as expected. Despite the three algorithms giving very similar results for the path and lattice networks, when applied to real networks the results are not as close. This suggests that path and lattice graphs may be 'too simple' to yield differences between the three algorithms. The other major issue is that we need a larger sampling pool to report the mean fractal dimension and standard deviations reliably. Also, for the power law behavior to emerge (see Equation 1), we need at least an order of magnitude change in the range of box size. This was impossible to obtain for some networks, like *E. coli*, or computationally costly for some algorithms, like the Box Burning method. The Burning method also demands relatively high RAM usage (approx. 20 GB for *E. coli*) which necessitated the use of a High Performance Computing cluster to generate some of the results on cellular networks.

The fuzzy method uses a completely different approach to the computation of networks' fractal dimension from the other two methods and therefore we can expect some deviations depending on the particular structure of the network. However, we expect the greedy algorithm and the burning algorithm to give similar results. Apart from *Human herpesvirus 8* this is not quite the case. This can be explained by the fact that the greedy algorithm uses a larger range of box lengths l_B (from 1 to the diameter of the network) while the burning method only uses boxes up to roughly half of the diameter. We expect that including a larger range of box lengths would improve the results from burning method, but this was not possible due to the computational complexity of the algorithm.

Table 2 also compares our results to other works in the literature. Given the high and increasing number of databases for biological networks, people tend to use different databases and different versions for PPINs. Only two of the results obtained by Wang *et al.* uses the same BioGRID database. The original paper by Song *et al.* (18) studies a metabolic network in *E. coli* (instead of a PPIN). This, combined with the use of different algorithms, makes the comparison with the literature quite challenging.

Materials and Methods

For a network $\mathcal{G}$ and box size l_B a compact box is defined as the set of nodes where all the distances l_{ij} between nodes i and j are smaller than l_B and N_B is the minimum number of boxes needed to cover the entire network $\mathcal{G}$ (30). As can be seen from Equation 1, the fractal dimension d of the network $\mathcal{G}$ is obtained by a linear fit of $log(N_B)$ versus $\log(l_B)$.

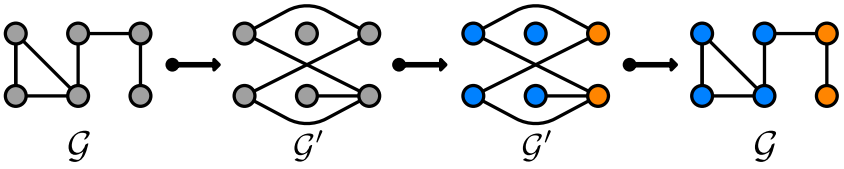


Fig. 8. Greedy coloring algorithm for box counting of Song *et al.* applied to a simple network for $l_B = 3$ (18). The algorithm starts with a network $\mathcal{G}$ and builds the dual network $\mathcal{G}'$ for a given box size l_B. A vertex coloring algorithm is applied to the dual network $\mathcal{G}'$ so that no connected vertices have the same color. The color scheme transferred to the original network gives its box covering. Figure adapted from Ref. (18).

In this section we will briefly review and describe the different methods we used to compute the minimum number of boxes N_B of size l_B needed to cover a given network $\mathcal{G}$.

Greedy Coloring Fractal Dimension. Song *et al.* showed that the problem of finding the minimum number of boxes N_B for a given box size l_B can be mapped to a coloring problem (18), a well known NP-hard problem in graph theory (31).

In order to compute $N_B(l_B)$ for a given network $\mathcal{G}$ using Song's greedy coloring algorithm, one has to compute the dual network $\mathcal{G}'$, defined as the network where the nodes of the original network $\mathcal{G}$ are connected if their distance in $\mathcal{G}$ is greater than or equal to l_B. Once the network $\mathcal{G}'$ is computed a vertex coloring algorithm gives the natural number of boxes as the number of colors (18). This is because vertex with the same colors in $\mathcal{G}'$ are at a distances smaller than or equal to l_B in the original network $\mathcal{G}$ and therefore are within the same box. This concept is illustrated in Figure 8 for a simple case ($N = 6$ nodes, $M = 6$ edges and $l_B = 3$).

The vertex coloring problem is an NP-hard problem and therefore a brute-force solution is computationally very expensive (32). However, a greedy coloring algorithm can give a good approximation of this problem in reasonable time (18). The drawbacks of this approach is that the results are dependent on the choice of the starting node, therefore some statistics has to be performed.

The naive implementation of Song's algorithm (calculation of all shortest paths, explicit construction of the dual network, greedy coloring) requires the construction of the dual network for every box length l_B, which is an $\mathcal{O}(N^2)$ operation. However, Song *et al.* suggest a very efficient implementation that only needs a single network pass (18).

Fuzzy Fractal Dimension. In order to overcome the limitations of the greedy coloring algorithm, Zhang *et al.* proposed a fuzzy method to calculate the inverse of the minimum number of boxes of a given size in order to cover a graph $\mathcal{G}$ (19). The idea behind this algorithm is to use spherical boxes that can actually overlap, in order to find the covering ability, i.e. the inverse of the minimum number of boxes, for a given box size l_B.

In the fuzzy fractal dimension method, balls of radius l_B are constructed around each node of the given network $\mathcal{G}$, and the inverse of the minimum number of boxes (the covering ability) needed to cover the graph for a given size is given by

$$N^{-1}(l_B) = \frac{1}{N(N-1)} \sum_{i}^{N} \sum_{j \neq i}^{N} \Omega_{ij}(l_B) A_{ij}(l_B)$$

where

$$\Omega_{ij}(l_B) = \begin{cases} 1, & d_{ij} \leq l_B \\ 0, & \text{otherwise} \end{cases} \qquad \text{and} \qquad A_{ij} = \exp\left(-\frac{d_{ij}^2}{l_B^2}\right).$$

As usual, d_{ij} is the shortest path between nodes i and j. Ω_{ij} is a connectivity factor describing whether the ball centered at node i can cover node j, and A_{ij} is a measure of how well the ball centered at node i can cover node j.

The algorithm suggested by Zhang *et al.* computes the coverage ability $N_i^{-1}(l_B) = \sum_{j \neq i} \Omega_{ij}(l_B) A_{ij}$ for every node i in the network $\mathcal{G}$ for a given l_B. The overall coverage ability is simply the average of such values (19). For the network of Figure 8 with $l_B = 3$, we have (19)

$$N_1^{-1}(3) = 0.594, \quad N_2^{-1}(3) = 0.594, \quad N_3^{-1}(3) = 0.739, \quad N_4^{-1}(3) = 0.743, \quad N_5^{-1}(3) = 0.633, \quad N_6^{-1}(3) = 0.448,$$

so that the final coverage ability for $l_B = 3$ is the average $N^{-1}(3) = 0.625$. This means that the number of boxes with size $l = 3$ needed to cover the network is $N(3) = 1.6$. This result is similar to the one obtained with the algorithm of Song *et al.* but the method is much easier to implement and is fully deterministic so that no averaging over multiple runs is needed.

Burning Algorithm. This method, first proposed in Reference (16), tiles the network with boxes of size l_B. All the nodes within a box should be 'reachable' by a distance smaller than the given l_B (i.e. the box should be compact). Once the compact box is formed, all the nodes inside are burned and the total number of boxes increase by 1. The same protocol continues until all the nodes in the network are removed.

(i) Traditional Compact-Box-Burning (CBB) algorithm. This method was first introduced by Song *et al.* (18) and can be summarized as follows: (1) Construct a set S of all yet uncovered nodes. (2) Choose a random node in S and remove it from the set. (3) Take all nodes whose distance from the initial node is less than the box size l_B and remove them from S (compactness of the box). (4) Repeat steps (2) and (3) until the set S is empty.

One of the drawbacks of this method is that to form each compact box, it has to go through all the nodes in the network and test whether they belong to the same box. This makes the code substantially slow. Furthermore, it starts by picking a random node in the network. As we discussed in the results section, this introduces random fluctuations in fractal dimension estimates which have to be averaged out by running the code several times to obtain a reliable result. For these two reasons, we propose a new method that addresses these issues.

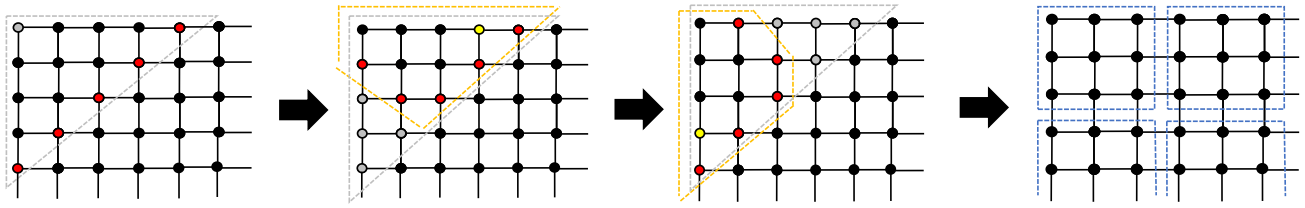


Fig. 9. A novel box-burning method. First, we randomly select a node with the least number of neighbors (yellow node). Then, find the local diameter nodes (red nodes) and form a box around all the nodes inside the local diameter (gray dashed line). Then, we go through each node inside the gray box and find all the nodes within the local diameter nodes (yellow dashed lines) and store the list. Finally, we take the intersection of all the yellow boxes and if it is bigger than the size of the minimum box, then we form the box and burn the nodes inside.

(ii) A Novel Box Burning Algorithm. This new approach that we developed aims at resolving the issues raised by the traditional CBB algorithm. We apply the following steps in computing the minimum number of boxes (see Figure 9): (1) Randomly select a node p out of all the ones that have the lowest number of immediate neighbors. (2) Find all the Local Diameter nodes (see Table 1) that are at distance $l_B - 1$ away from p. Find all the nodes that are in the shortest path from p to the Local Diameter nodes and store them in a set S. (3) For each node i in S, find all the neighboring nodes s'_i that are at distance $l_i \leq l_B - 1$ away from i and store them in a set $s'_i \in S'$. (4) Find a set $s^* \in S'$ with the least number of nodes and define it to be the 'Minimum Box'. (5) Find the intersection over all elements s'_i in S'. If that has more nodes than the Minimum Box, remove all of them from the network and go back to step (1). If the intersection has fewer nodes than the Minimum Box, remove one node from the Local Diameter of node p and go back to step (2). Continue until all the nodes are removed.

ACKNOWLEDGMENTS. The authors thank Dr. Florian Klimm for fruitful discussions about the project.

Feedback

Working on this project we learned about PPINs and how graph theory can be applied to such networks in order to better understand their properties. This is an intriguing field and the project was entertaining while being challenging. However, we found that the literature is somewhat confusing, with different authors using different methods and datasets so that a comparison within the field is not always trivial.

During this project we spent some time learning new concepts in programming and software engineering. We realized how version control helps collaborating on a code base on GitHub (github.com) and tried to apply some software engineering practices. In particular, we used the Python test framework `pytest` and we set up continuous integration using Travis-CI (travis-ci.org) so that the test suite is run for every commit. We also set up code coverage on CodeCov (codecov.io), which helped to spot missing tests and dummy code.

1. Kramer T, Lo Monte F, Goring S, Okala Amombo GM, Schmidt B (2012) Small molecule kinase inhibitors for lrrk2 and their application to parkinson's disease models. *ACS chemical neuroscience* 3(3):151–160.
2. Liu H, et al. (2011) Prioritizing cancer-related genes with aberrant methylation based on a weighted protein-protein interaction network. *BMC systems biology* 5(1):158.
3. Oti M, Snel B, Huynen MA, Brunner H (2006) Predicting disease genes using protein-protein interactions. *Journal of medical genetics* 43:691–698.
4. Ivanov AA, Khuri FR, Fu H (2013) Targeting protein-protein interactions as an anticancer strategy. *Trends in Pharmacological Sciences* 34:393–400.
5. Laraia L, McKenzie G, Spring DR, Venkitaraman AR, Huggins DJ (2015) Overcoming chemical, biological, and computational challenges in the development of inhibitors targeting protein-protein interactions. *Chemistry & Biology* 22:689–703.
6. Phizicky EM, Fields S (1995) Protein-protein interactions: methods for detection and analysis. *Microbiological Reviews* 59:94–123.
7. Rao VS, Srinivas K, Sujini G, Kumar GNS (2014) Protein-protein interaction detection: Methods and analysis. *International Journal of Proteomics* 2014:147648.
8. Fields S, Song O (1989) A novel genetic system to detect protein-protein interactions. *Nature* 340:245–246.
9. Chatr-aryamontri A, et al. (2017) The biogrid interaction database: 2017 update. *Nucleic Acids Research* 45(Database-Issue):D369–D379.
10. Yates JR, Ruse CI, Nakorchevsky A (2009) Proteomics by mass spectrometry: Approaches, advances, and applications. *Annual Review of Biomedical Engineering* 11:49–79.
11. Pedamallu C, Ozdamar L (2014) A review on protein-protein interaction network databases. *Springer Proceedings in Mathematics & Statistics* 73:511–519.
12. EMBL-EBI (2018) Network analysis in biology (https://www.ebi.ac.uk/training/online/course/network-analysis-protein-interaction-data-introduction/network-analysis-biology-0).
13. Zhu D, Qin ZS (2005) Structural comparison of metabolic networks in selected single cell organisms. *BMC bioinformatics* 6(1):8.
14. Bergmann S, Ihmels J, Barkai N (2003) Similarities and differences in genome-wide expression data of six organisms. *PLoS biology* 2(1):e9.
15. Wuchty S, Barabási A, Ferdig MT (2006) Stable evolutionary signal in a yeast protein interaction network. *BMC evolutionary biology* 6(1):8.
16. Song C, Havlin S, Makse HA (2005) Self-similarity of complex networks. *Nature* 433:392–395.
17. Concas G, Locci M, Marchesi M, Pinna S, Turnu I (2006) Fractal dimension in software networks. *Europhysics Letters* 76:1221–1227.
18. Song C, Gallos LK, Havlin S, Makse HA (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.
19. Zhang H, Hu Y, Lan X, Mahadevan S, Deng Y (2014) Fuzzy fractal dimension of complex networks. *Applied Soft Computing* 25:514–518.
20. Grolmusz V (2015) A note on the pagerank of undirected graphs. *Information Processing Letters* 115(6):633 – 634.
21. Na CH, et al. (2012) Synaptic protein ubiquitination in rat brain revealed by antibody-based ubiquitome analysis. *Journal of Proteome Research* 11(9):4722–4732. PMID: 22871113.
22. Marinovic AC, Zheng B, Mitch WE, Price SR (2002) Ubiquitin (UbC) expression in muscle cells is increased by glucocorticoids through a mechanism involving Sp1 and MEK1. *Journal of Biological Chemistry* 277(19):16673–16681.
23. Gack MU, et al. (2007) TRIM25 RING-finger E3 ubiquitin ligase is essential for RIG-I-mediated antiviral activity. *Nature* 446(7138):916–920.
24. Harrington AW, et al. (2011) Recruitment of actin modifiers to TrkA endosomes governs retrograde NGF signaling and survival. *Cell* 146(3):421–434.
25. Shafiee G, Asgari Y, Soltani A, Larijani B, Heshmat R (2018) Identification of candidate genes and proteins in aging skeletal muscle (sarcopenia) using gene expression and structural analysis. *PeerJ* 6:e5239.
26. Kittel C (2004) *Introduction to Solid State Physics*. (John Wiley & Sons).
27. Wang D, Yu Z, Anh V (2011) Self-similarity in weighted ppi networks. *International Conference on Bioscience, Biochemistry and Bioinformatics* 5:193–197.
28. Wei D, et al. (2013) Box-covering algorithm for fractal dimension of weighted networks. *Scientific Reports* 3:3049.
29. Jin Y, Turaev D, Weinmaier T, Rattei T, Makse HA (2013) The evolutionary dynamics of protein-protein interaction networks inferred from the reconstruction of ancient networks. *PLOS one* 8:e58134.
30. Feder J (1988) *Fractals*. (Springer US).
31. Garey M, Johnson D (1979) *Computers and Intractability: A Guide to the Theory of NP-Completeness*. (Freeman).
32. Christofides N (1971) An algorithm for the chromatic number of a graph. *The Computer Journal* 14:38–39.