Graphlets and motifs

Maša Kljun[a], **Jan Krivec**[a], **and Matija Teršek**[a]

[a]University of Ljubljana, Faculty of Computer and Information Science, Večna pot 113, SI-1000 Ljubljana, Slovenia

The manuscript was compiled on May 28, 2022

Almost any system, let it be biological, technological, or social, can be represented in a form of a network. To learn possible interactions and underlying concepts of these networks, we can analyze them on different levels. For example, we can analyze networks on a link level by considering links as building blocks of networks. However, analysis on such a level is sometimes too fine-grained. An alternative approach is to characterize and discriminate networks with different kinds of building blocks – sub-graphs or fragments. Depending on whether these sub-graphs are induced or not, we can further divide them into two groups – graphlets and motifs. In this project, we discuss the motivation and history of using graphlets and motifs in network analysis. We present the concepts of graphlets and motifs and provide the mathematical theory that is required to understand these concepts. Furthermore, we show how we can use graphlets and motifs in practice through a few short practical examples.

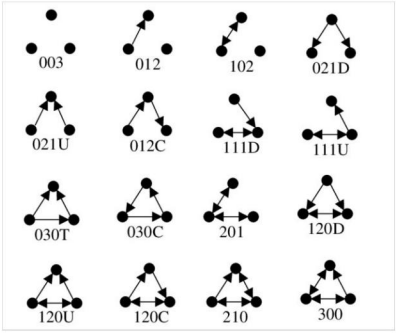


Fig. 1. The 16 isomorphism classes for digraphs with 3 nodes. These graphs represent triad census, which is a predecessor of motifs. By omitting 003, 012, and 102 we get all 13 directed three-node motifs.Triad naming convention: the first digit represents the number of mutual diads, the second digit represents the number of asymmetric diads, while the third number represents the number of null diads. Further letters differentiate among triad types.

Problem definition, motivation, and background. Network analysis has many useful applications in fields such as biology, sociology, etc., as a variety of real-world systems can be represented in a form of a network. By analyzing networks we are learning the interactions within the systems which could otherwise stay undiscovered. Different approaches for studying networks have been proposed, including looking at the networks from a subgraph perspective. By doing that we are looking at the network's basic structural elements and identifying underlying patterns.

Analyzing networks by studying their subgraphs dates back to 1974, when Holland and Leinhardt (1) introduced the triad census. It is defined as a vector of 16 components, that represent 16 isomorphism classes for digraphs with 3 nodes (see Figure 1). They can be used for the examination of local structural properties by comparing their count distributions across different real and random networks.

Milo et al. (2) further generalize the idea and define network motifs, as patterns of interconnections occurring in complex networks at numbers that are significantly higher than those in randomized networks. They show that motifs vary across networks from different domains and that we can use them to differentiate between biology, neuronal, information, social, and language networks. But to classify a subgraph as a certain motif, we need to investigate its frequency in a random graph and compare it to the frequency in the investigated graph.

Because this is a costly process and not a trivial task, Pržulj et al. (3) define graphlets - small connected and induced subgraphs of a given simple undirected graph. Graphlets can be used to characterize local neighborhoods around nodes, instead of on a network level as motifs. They have been widely utilized in biology (4–6), computer networking (7–9), chemoinformatics (10), and for image segmentation (11). Graphlets and motifs are used in network comparisons, as they have proven to be effective in network characterization and discrimination, and are a popular research field (12). In the literature, people often use the terms graphlets and motifs interchangeably, however, in the following sections we define them as in the original papers, where they were introduced. Additionally, we explain the differences between them.

The most frequent use of graphlets and motifs is for the network analysis. For that reason, we further explore mathematical concepts which will help us define graphlets and motifs more precisely and show how they can be useful for network analysis.

1. Mathematical theory

In this chapter, we assume that the reader is familiar with network concepts and basic terminology of graphs (e.g., degree of a node) and some mathematical concepts (e.g., bijection).

Theorem 1.1 (Subgraph) *We can define subgraph $G_s = (V_s, E_s)$ (with vertices V_s and edges E_s) of a graph $G = (V, E)$ (with vertices V and edges E) as a graph, for which $V_s \subseteq V$ and $E_s \subseteq E$.*

Theorem 1.2 (Fragment) *We say a subgraph $G_s = (V_s, E_s)$ is connected, if $\forall v, u, v \in V_s, u \in V_s$ there exists a path between them. Connected subgraphs are also called fragments.*

Following the definition of a subgraph, we can also define a notion of the induced subgraph, which is just a special case of a subgraph.

Theorem 1.3 (Induced subgraph) *A subgraph G_s with vertices V_s and edges E_s of graph G with vertices V and edges E is said to be induced, if and only if $\forall(u,v) \in E_s \iff (u,v) \in E$.*

The definition of an induced subgraph states that two vertices u, v in G_s are adjacent in G_s if and only if they are adjacent in G, or in other words G_s, has the same edges as G between vertices of G_s. We show an example of a graph, subgraph, and induced subgraph in Figure 2.

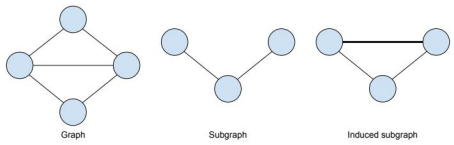


Fig. 2. Top row shows the original graph. Second row shows a subgraph, but it is not induced, as edge between the left and the right node is missing. Bottom row shows the induced subgraph.

Now we are familiar with the definitions of a subgraph and induced subgraph, but to present the concept of graphlets and motifs, we also need to define the isomorphism and automorphism of a graph.

Theorem 1.4 (Isomorphism) *An isomorphism of graphs G_1 and G_2 is a bijection between the sets of vertices V_{G_1} and V_{G_2} denoted as $f : V_{G_1} \to V_{G_2}$, where adjacency of vertices is preserved, i.e., if edge $\{u,v\} \in G_1$, then $\{f(u), f(v)\} \in G_2$. Furthermore, the number of links in the isomorphic graph G_2 must be the same as in the original G_1.*

Theorem 1.5 (Automorphism) *An automorphism of a graph $G = (V,E)$ is a permutation of $\sigma : V \to V$, such that the pair of vertices (u,v) form and edge $\iff$ $(\sigma(u), \sigma(v))$ also form an edge. In other words, it is graph isomorphism from G to itself. Automorphisms of graph G form a group called the automorphism group of G denoted by $Aut(G)$.*

Informally, graphs are isomorphic if they have the same edge structure (the same topology) if we ignore the distinction between individual nodes. We show an example in Figure 3. We now have enough mathematical knowledge and background to define motifs and graphlets, and explore how to use them in network analysis.

2. Motifs

Definition 2.1 (Motif) *Motifs are recurring patterns of interconnections in the network. More precisely, they are frequent non-induced fragments.*

Motifs can be composed of different amounts of nodes. Motifs can be undirected or directed subgraphs. We can see all the examples of directed three-node motifs in Figure 4.

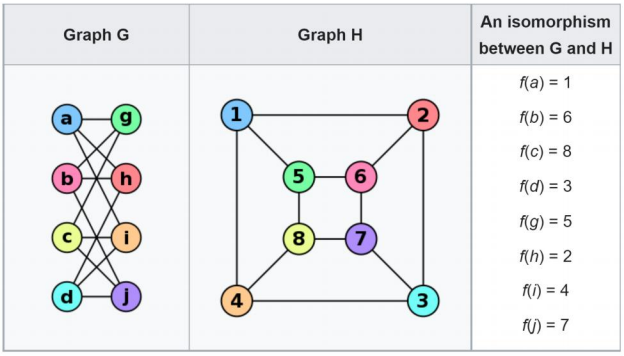


Fig. 3. Figure shows original graph G and graph H that is isomorphic to G. We see that both graphs have the same edge structure or the same topology, if we ignore the distinction between individual nodes. Source: Wikipedia.

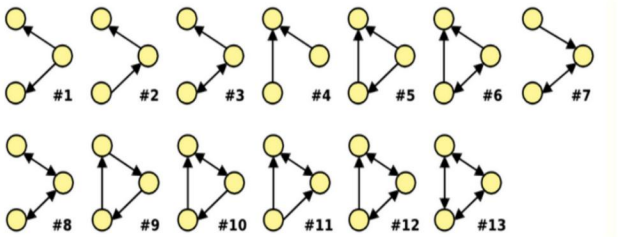


Fig. 4. All motifs composed of 3 nodes on directed graphs. Source (13).

They frequently appear in real networks. Giant networks are very hard to analyze and here motifs prove to be particularly relevant when describing the architecture of real-life networks. Motifs with a smaller amount of nodes can be represented as building blocks of large networks and can help us understand how networks work. Some motifs (usually those with fewer nodes) appear more often in networks than others. When looking for motifs, we allow them to overlap.

We can measure the significance of a motif based on how many times it appears in the network. We define motif significance as the function of expectation for a motif to occur in a graph, given its size. First, we generate a random graph in such a way that it has similar basic characteristics as the original, which we use to compute the significance.

Definition 2.2 (Significance) *Significance is defined as $Z_i = \frac{N_i^{real} - \langle N_i^{rand} \rangle}{std(N_i^{rand})}$, where N_i^{real} represents the number of motifs of type i in our real network and N_i^{rand} represents the expected number of motifs of type i in the random graph (14).*

As we can see, significance is defined as a normalized z-score. We can also define Network significance profile, which can serve as a oracle for comparing networks. We show an example in Figure 8 where we can see which motifs are over or under-represented in different types of networks. By only inspecting the plots, we can already differentiate some network types from others as some motifs are over-represented in one type of network, while under-represented in other types.

Definition 2.3 (Network significance profile) *Network significance profile (SP) is defined as a vector of normalized Z-scores* $SP_i = \frac{Z_i}{\sqrt{\sum_j Z_j^2}}$.

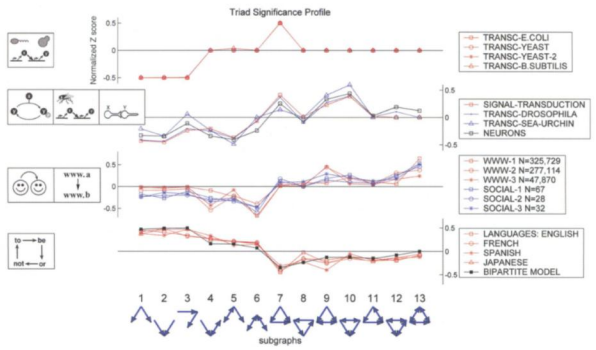


Fig. 5. Significance profile of size-3 motifs in four different types of networks (y-axis shows significance profile (SP) for each network). We can see that the same types of networks have similar SP (e.g., social networks have similar SP). Source: Milo et al. (2).

To compute a representative value for significance scores, we have to generate an appropriate random graph. The optimal solution is generating a random graph, that has the same number of nodes and edges as the original graph and also the same degree sequence as the original. For this purpose, we can use different methods.

1. Configuration model. With a given degree sequence, we can represent vertices as stubs (15). In every iteration, we choose two random stubs and connect them, until no stubs are left. The end result is a random graph that very closely resembles the original. We can ignore double edges and self loops because, on a large real-life network, these omitted links will represent small noise. We show an example in Figure 6.

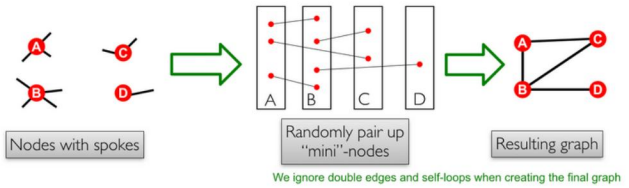


Fig. 6. Example of constructing a random graph using a configuration model. Source: (13)

2. Switching (Rewiring). We start with a given graph G. In each iteration we choose two pairs of connected nodes (A → B, C → D) and we switch the endpoints (A → D, C → B). We repeat this $Q \cdot |E|$, where Q needs to be large enough for the process to converge. This process is much slower than the Configuration model.

We can use another measure called abundance ratio, which is less dependent on network size.

Definition 2.4 (Abundance ratio) *Abundance ratio is defined as* $\delta_i = \frac{N_i^{real} - \langle N_i^{rand} \rangle}{N_i^{real} + \langle N_i^{rand} \rangle + \epsilon}$, *where* N_i^{rand} *and* N_i^{real} *are defined as in definition 2.2 and* ϵ *ensures that* $|\delta_i|$ *is not misleadingly large, when there are very few appearances of a certain subgraph in both real and random graph (2).*

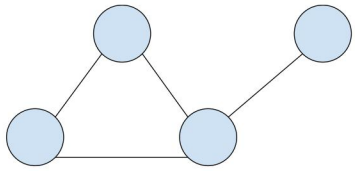
Similar to how significance profile is defined using significance, the abundance ratio is used in abundance/ratio profile.

Definition 2.5 (Abundance/ratio profile) *The subgraph abundance/ratio profile (SRP) is a normalized vector of abundance ratios* δ_i *for each motif* i $SRP_i = \frac{\delta_i}{\sqrt{\sum \delta_i^2}}$

3. Graphlets

Definition 3.1 (Graphlet) *Graphlets are non-isomorphic, induced fragments (3).*

We show the difference in counting graphlets and motifs in Figure 7. We see that we have only one triangle and one connected triplet when talking about graphlets (because they are induced and we have to consider all edges between three nodes), while we have 5 linked triplets. Motifs are not induced, and we can drop edges when counting them.



	Graphlets	Motifs
	1	1
	1	5

Fig. 7. Identifying graphlets and motifs on the same graph.

In Figure 10 we show all possible graphlets with 2, 3, 4, and 5 nodes. Notice that with an increasing number of nodes n, the number of graphlets rapidly increases. Simply by counting the number of graphlet occurrences in a network, we can define relative graphlet frequency F_i.

Definition 3.2 (Relative graphlet frequency) *Relative graphlet frequency* $F = \{F_i | i \text{ graphlet type}\}$, *where* $F_i = \frac{n_i}{\sum_i n_i}$, *for number* n_i *of graphlets* i *in real network (3).*

Consequently, we can define a distance between two graphs

G_1 and G_2 based on relative graphlet frequency (3), as

$$D(G_1, G_2) = \sum_i |F_i(G_1) - F_i(G_2)|, \qquad [1]$$

which shows that we can easily use relative graphlet frequency to compare the networks. In Figure 8 we show relative graphlet frequencies for protein-protein interaction (PPI) networks and four random networks from (3). We can see that random networks have similar behaviour and relative frequencies, while F of PPI is different.

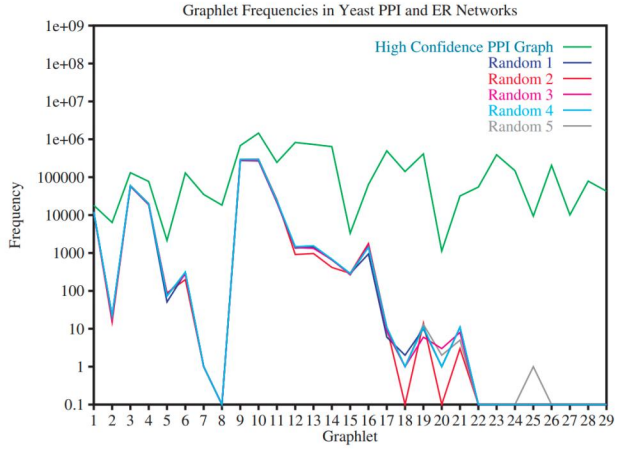


Fig. 8. Graphlet frequencies of PPI and random graphs. We see that we can use relative graphlet frequenciy to discern between graphs. Source: Pržulj et al. (3)

We can also use graphlets for network analysis on a node level. More specifically, we use graphlet degree distribution (GDD) to measure the local structural properties of a network. To define graphlet degree distribution, we first have to define the automorphism orbit.

Definition 3.3 (Automorphism orbit) *Let $v \in V$ a node from graph G, and $Aut(G)$ be the automorphism group of graph G. Then we can define automorphism orbit of V as $Orb(v) = \{u \in V | u = g(v)$ for some $g \in Aut(G)\}$ (16).*

In Figure 10 we also show the automorphism orbits. For example, G_0 has only one automorphism orbit. That is because we have only one non-isomorphic position, meaning whether we position ourselves at the top or the bottom node, we are both times at the beginning of a chain. Similarly, in G_2 we can be at any of the three corner nodes, which always represent the same position inside the triangle due to symmetry. To understand graphlets with more than one automorphism orbit, we can take a look at graphlet G_1, where the top and the bottom black nodes denote one position (beginning of the chain), while the white node denotes the middle position. Similarly, orbits are shown and numbered for other graphlets.

Notice that the graphlet with 2 nodes (G_0) is an edge. Furthermore, to obtain a degree distribution of a network, we count how many nodes touch one G_0, how many nodes touch two G_0, and so on. Nothing prevents us from using the same approach for other graphlets (G_1–G_{29}), or more specifically for other 72 automorphism orbits on these graphlets. For example, we count how many nodes touch one triangle (graphlet G_2), how many touch two triangles, and so on. As seen, for graphlet G_1 we have two orbits, so we can touch graphlet G_1 at two positions. Thus, we count how many nodes touch one G_1 at the end node (orbit 1), we count how many nodes touch two G_1 at the end, ..., but we also count how many nodes touch one G_1 at the mid node (orbit 2), how many nodes touch two G_1 at the mid node, and so on. Analogous to the degree distribution, we obtain 73 distributions, one for each orbit. We denote these distributions as p_k^i, where i represents an orbit.

Definition 3.4 (i-th orbit GDD) *We define i-th orbit graphlet degree distribution $p_k^i = \frac{N_k^i}{\sum_j N_j^i}$, where N_k^i represent how many nodes touch orbit i k-times. Naturally, it holds $\sum_k p_k^i = 1$.*

Definition 3.5 (i-th orbit scaled GDD) *For i-th graphlet degree distribution p_k^i, i-th orbit scaled degree distribution is defined as $\tilde{p}_k^i = p_k^i / k$. (16)*

Note that p_k^0 thus denotes the network's degree distribution. We see that scaled distribution is simply scaled by degree k, thus we can use it when we want to decrease the contribution of larger degrees.

Having defined the 73 graphlet degree distributions, we can now define a network agreement measure. We do this as direct comparison of 73 direct distributions is not intuitive. Network agreement measure acts as a similarity measure denoting how similar are the two networks.

Definition 3.6 (i-th orbit graphlet agreement) *We define i-th orbit graphlet agreement $A_i = 1 - \sqrt{\frac{1}{2}\sum_k (\tilde{q}_k^i - \tilde{p}_k^i)^2}$, where $\tilde{q}_k^i$ and $\tilde{p}_k^i$ are i-th orbit graphlet degree distributions of network G_1 and G_2, respectively.*

Definition 3.7 (graphlet agreement) *We define graphlet agreement between graphs G_1 and G_2 as arithmetic or geometric mean of i-th orbit graphlet agreements A_i (16):*

$$A = \frac{1}{73}\sum_i A_i \qquad \textit{(arithmetic mean)}$$
$$A = (\Pi_i A_i)^{\frac{1}{73}} \qquad \textit{(geometric mean)} \qquad [2]$$

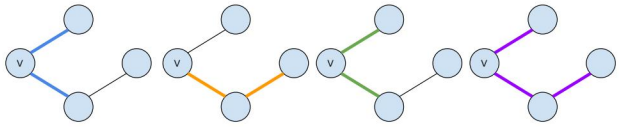
We see that graphlet agreement is simply arithmetic or geometric mean of i-th orbit graphlet agreements, and each i-th orbit graphlet agreement is similarity between i-th orbit graphlet degree distributions.

We can also take a different approach, and compute a graphlet degree vector (GDV) with 73 coordinates corresponding to each of the automorphism orbits for graphlets with 2 to 5 nodes. By that we get a signature of a node, describing the topology of the node's neighborhood. In this particular case, we are describing the structure of a network around the node at a distance of at most 4 hops*.

*We are considering graphlets up to 5 nodes

Definition 3.8 (graphlet degree vector) *Graphlet degree vector of node v is defined as $GDV(v) = \{g_i^v\}$, where g_i^v is number of graphlets that touch v at orbit i (17).*

Graphlet degree vector provides a measure of a node's local network topology. We can compare two graphlet degree vectors and obtain a highly constraining measure of local topological similarity between them. Use case of graphlet degree vectors in biology is to compute GDV for each node, and by that identify groups of topologically similar nodes. Furthermore, GDVs can be used to predict the biological properties of yet uncharacterized nodes based on known biological properties. See Figure 9 to see how GDV is calculated.



Orbit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
GDV	2	1	1	0	0	1	0	0	0	0	0	0	0	0	0

Fig. 9. Example calculation of GDV. We have a simple undirected graph in which we count orbits 0-14. Each replica of a graph above shows the orbits as colored edges. We see that there are two 0-orbits, and one 1, 2, and 5 orbit.

4. Finding motifs and graphlets in networks

Finding out if a subgraph exists in the original graph is computationally a hard problem. Complexity also grows with the size of the subgraph, so the search is usually limited to subgraphs of sizes from 3 to 8 nodes. Finding k-motifs and graphlets requires us to:

1. Enumerate all size-k connected subgraphs.
2. Count the number of occurrences of each subgraph type (with graph isomorphism test)

Determining, whether two subgraphs are isomorphic is an NP-complete problem, which adds to computational complexity. There are several algorithms for finding subgraphs:

- Exact Subgraph Enumeration (ESU) (18)
- Kavosh Algorithm (19)
- Subgraph Sampling (20)
- Orbit Counting Algorithm (Orca) (21)

We will take a closer look at ESU algorithm, which is a baseline approach for subgraph counting. We will keep track of two sets:

- V_{sub}, which holds currenty construted subgraph (motif).
- V_{ext}, which holds a set of candidate nodes to extend the subgraph.

The main idea is to start with a single node v, and next add the nodes u to V_{ext} if two conditions hold:

1. ID of a node u has to be larger than that of v
2. u may only be neighbored to some newly added node w, but not to any node already in V_{sub} (we want to only extend the subgraph that we have built so far).

ESU is a recursive algorithm and it builds an implicit tree-like structure of maximal depth k (size of subgraph). We call this structure the **ESU-Tree**. Pseudocode for the algorithm:

Algorithm 1 Exact Subgraph Enumeration (ESU)

Input graph G = (V, E), integer k ($1 \leq$ k $\leq$ |V|)
Output All size-k subgraphs in G

```
EnumerateSubgraphs(G, k):
for each vertex v ∈ V do
    ▷ extend V_ext with all neighbouring nodes u, that have a higher id than vertex v
    V_ext ← {u ∈ N({v}) : u > v}
    ExtendSubgraph({v}, V_ext, v))
return
```

```
ExtendSubgraph(V_sub, V_ext, v):
if |V_sub| = k then
    return G[V_sub]
while V_ext ≠ ∅ do
    Remove arbitrary chosen vertex w from V_ext
    ▷ extend V_ext with all vertices u from exclusive neighbourhood, that have a higher id than vertex v
    V'_ext ← V_ext ∪ {u ∈ N_excl(w, V_sub) : u > v}
    ExtendSubgraph(V_sub ∪ w, V'_ext, v)
return
```

$N_{excl}(w, V_{sub})$ denotes the exclusive neighbourhood, e.g. all nodes neighbouring w, but not in V_{sub} or $N(V_{sub})$ (13).

As we can see, ESU is a simple algorithm, but it is also really slow and computationally expensive. The fastest algorithm out of those stated is Orca. We can see the comparison between Orca and a few other algorithms in Figure 11. As we can see from the results, when using larger graphlet sizes on really big networks, Orca is the only algorithm, that finished computation.

Network	Four-node graphlets				Five-node graphlets		
	FANMOD	GraphCrunch	RAGE	Orca	FANMOD	GraphCrunch	Orca
S.cerevisiae	62 s	4.4 s	1.7 s	<0.1 s	87 min	9.5 min	6.6 s
E.coli	34 s	1.8 s	1.0 s	<0.1 s	38 min	4.1 min	4.8 s
D.melanogaster	21 s	3.1 s	1.6 s	<0.1 s	18 min	2.8 min	2.3 s
Human	/	183 min	11.8 min	6.1 s	/	/	269 min
Internet autonomous systems	574 min	37 min	3.0 min	2.5 s	/	/	49 min

Note: We aborted the algorithms that took more than a day and marked the corresponding results with /.

Fig. 11. Comparison of algorithms on real-world networks. FANDOM algorithm represents the ESU algorithm. Source: (21).

A. Practical example. We can use Python *netsci* library (22) for finding motifs. *Netsci* can only find motifs with three node. We will use *Networkx* for reading the network. We used the bottlenose dolphins network. Although this is an undirected network, we will present it as directed, to differentiate between more types of motifs.

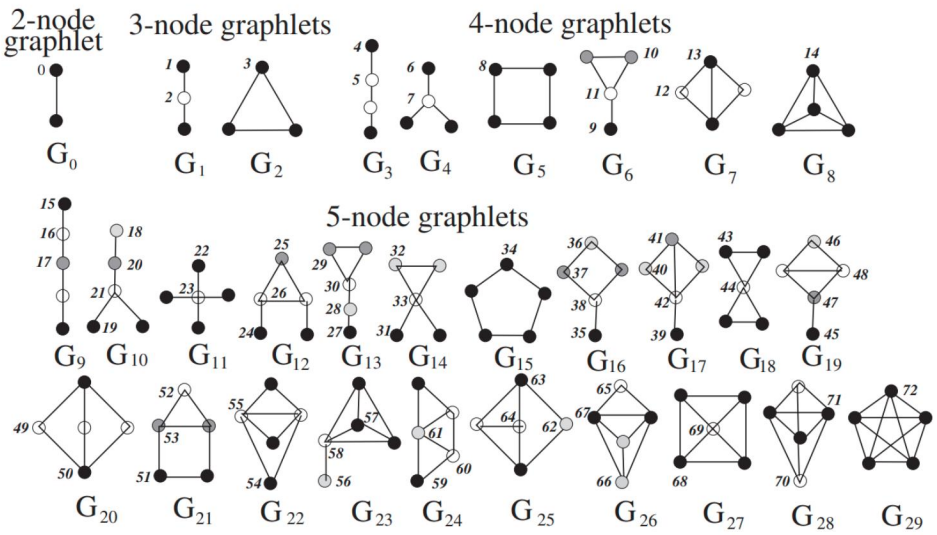


Fig. 10. All 2-node, 3-node, 4-node, and 5-node graphlets. They are numbered from 1 to 29. The small numbers beside nodes denote the automorphism orbits. Source: Pržulj et al. (3)

```
import networkx as nx
import netsci.metrics.motifs as nsm

graph = nx.read_adjlist('dolphins.net',
                        create_using=nx.DiGraph)
A = nx.adjacency_matrix(graph)

all_motifs = nsm.motifs(A)
print(all_motifs)

# [-1, -1, -1, 455, 1398, 413, ...
# ... 238, 2, 48, 0, 0, 0, 0, 12, 0, 0]
```

The returned array represents the number of occurrences for each type of motif. As we can see from the result first three numbers in the array represent non-connected motifs. The rest 13 numbers represent the number of occurrences for each type.

To find out which array index belongs to which type of motif we can again use *netsci*:

```
import netsci.visualization as nsv

nsv.bar_motifs(all_motifs)
```

This function prints out a graph, which can be seen in Figure 12.

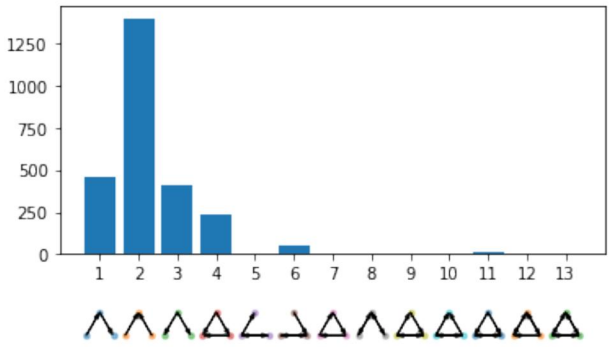


Fig. 12. Visualisation of motifs found using *netsci* library. We can see that for example motif 2 is much more represented than all other motifs, while some motifs do not even occur.

Discussion

In this paper, we presented a popular approach of analyzing networks – by using graphlets and motifs. For each of these 2 concepts, we provided the definitions and mathematical background required for understanding the possible use of graphlets and motifs in network analysis. We supported the definitions with simple examples and/or visualizations in order for this paper to serve as a concise overview of graphlets and motifs and a learning tool for future generations. The issue with using graphlets and motifs is that there is no appropriate up-to-date library in a popular programming language (e.g., Python), which could be used for finding more complex and larger graphlets and motifs.

1. Paul W Holland and Samuel Leinhardt. The statistical analysis of local structure in social networks, 1974.
2. Ron Milo, Shalev Itzkovitz, Nadav Kashtan, Reuven Levitt, Shai Shen-Orr, Inbal Ayzenshtat, Michal Sheffer, and Uri Alon. Superfamilies of evolved and designed networks. *Science*, 303 (5663):1538–1542, 2004.
3. Natasa Pržulj, Derek G Corneil, and Igor Jurisica. Modeling interactome: scale-free or geometric? *Bioinformatics*, 20(18):3508–3515, 2004.
4. Nino Shervashidze, SVN Vishwanathan, Tobias Petri, Kurt Mehlhorn, and Karsten Borgwardt. Efficient graphlet kernels for large graph comparison. In *Artificial intelligence and statistics*, pages 488–495. PMLR, 2009.
5. Oleksii Kuchaiev, Tijana Milenkovi'c, Vesna Memiševi'c, Wayne Hayes, and Nataša Pržulj. Topological network alignment uncovers biological function and phylogeny. *Journal of the Royal Society Interface*, 7(50):1341–1354, 2010.
6. Tijana Milenkovi'c, Weng Leong Ng, Wayne Hayes, and Nataša Pržulj. Optimal network alignment with graphlet degree vectors. *Cancer informatics*, 9:CIN–S4744, 2010.
7. David Hales and Stefano Arteconi. Motifs in evolving cooperative networks look like protein structure networks. *Networks Heterogeneous Media*, 3(2):239, 2008.
8. Dima Feldman and Yuval Shavitt. Automatic large scale generation of internet pop level maps. In *IEEE GLOBECOM 2008-2008 IEEE Global Telecommunications Conference*, pages 1–6. IEEE, 2008.
9. Luca Becchetti, Paolo Boldi, Carlos Castillo, and Aristides Gionis. Efficient semi-streaming algorithms for local triangle counting in massive graphs. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 16–24, 2008.
10. Hisashi Kashima, Hiroto Saigo, Masahiro Hattori, and Koji Tsuda. Graph kernels for chemoinformatics. In *Chemoinformatics and advanced machine learning perspectives: complex computational methods and collaborative techniques*, pages 1–15. IGI global, 2011.
11. Luming Zhang, Mingli Song, Zicheng Liu, Xiao Liu, Jiajun Bu, and Chun Chen. Probabilistic graphlet cut: Exploiting spatial structure cue for weakly supervised image segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1908–1915, 2013.

12. Shuo Yu, Yufan Feng, Da Zhang, Hayat Dino Bedru, Bo Xu, and Feng Xia. Motif discovery in networks: a survey. *Computer Science Review*, 37:100267, 2020.
13. Motifs and structral rules in network. https://snap-stanford.github.io/cs224w-notes/preliminaries/motifs-and-structral-roles_lecture. Accessed: 16.5.2022.
14. Sergi Valverde and Ricard V Solé. Network motifs in computational graphs: A case study in software architecture. *Physical Review E*, 72(2):026107, 2005.
15. Bailey K. Fosdick, Daniel B. Larremore, Joel Nishimura, and Johan Ugander. Configuring random graph models with fixed degree sequences. *SIAM Review*, 60(2):315–355, 2018. . URL https://doi.org/10.1137/16M1087175.
16. Nataša Pržulj. Biological network comparison using graphlet degree distribution. *Bioinformatics*, 23(2):e177–e183, 2007.
17. Tijana Milenković and Nataša Pržulj. Uncovering biological network function via graphlet degree signatures. *Cancer informatics*, 6:CIN–S680, 2008.
18. Sebastian Wernicke and Florian Rasche. FANMOD: a tool for fast network motif detection. *Bioinformatics*, 22(9):1152–1153, 02 2006. ISSN 1367-4803. . URL https://doi.org/10.1093/bioinformatics/btl038.
19. Zahra Kashani, Hayedeh Ahrabian, Elahe Elahi, Abbas Nowzari, Elnaz Saberi Ansari, Sahar Asadi, Shahin Mohammadi, and Falk Schreiber. Kavosh: A new algorithm for finding network motifs. *BMC bioinformatics*, 10:318, 10 2009. .
20. N. Kashtan, S. Itzkovitz, R. Milo, and U. Alon. Efficient sampling algorithm for estimating subgraph concentrations and detecting network motifs. *Bioinformatics*, 20(11):1746–1758, 03 2004. ISSN 1367-4803. . URL https://doi.org/10.1093/bioinformatics/bth163.
21. Tomaž Hočevar and Janez Demšar. A combinatorial approach to graphlet counting. *Bioinformatics*, 30(4):559–565, 12 2014. ISSN 1367-4803. . URL https://doi.org/10.1093/bioinformatics/btt717.
22.