

Network motifs detection using random networks with prescribed subgraph frequencies

Miguel E.P. Silva, Pedro Paredes, Pedro Ribeiro

CRACS & INESC-TEC

DCC-FCUP, Universidade do Porto, Portugal

mepsilva@dcc.fc.up.pt, pparedes@dcc.fc.up.pt, pribeiro@dcc.fc.up.pt

Abstract. In order to detect network motifs we need to evaluate the exceptionality of subgraphs in a given network. This is usually done by comparing subgraph frequencies on both the original and an ensemble of random networks keeping certain structural properties. The classical null model implies preserving the degree sequence. In this paper our focus is on a richer model that approximately fixes the frequency of subgraphs of size $K - 1$ to compute motifs of size K . We propose a method for generating random graphs under this model, and we provide algorithms for its efficient computation. We show empirical results of our proposed methodology on neurobiological networks, showcasing its efficiency and its differences when comparing to the traditional null model.

Keywords: Network Motifs, Random Graphs, Subgraph Counting

1 Introduction

Complex networks have been established as essential tools to model and analyze several real-life systems and problems. A technique that greatly contributed for this reputation is network motif analysis [15]. Network motifs consist of over-represented substructures of a network, or subgraphs that appear in a higher number than expected. This method has been used successfully in many fields of science, such as biology [22, 23] or sociology [4].

In order to perform a meaningful network motif analysis, it is important to decide on a definition of what is the expected frequency of a certain subgraph. To do so, one chooses a determined null model of random graphs and computes the average frequency of the given subgraph on this null model. The most used null model is maintaining the degree sequence of the original network [4, 13, 14, 23]. Other models have been proposed [3, 15], but here we focus on a new model.

One can think of graph edges as subgraphs of size 2. A natural extension would therefore be to maintain counts of larger subgraphs. Moreover, certain patterns can be essentially the consequence of over-represented smaller subgraphs contained in them. With all of this in mind we propose to keep the frequency of subgraphs of size $K - 1$ when discovering motifs of size K , aiming towards a much richer null model, able to really distinguish when a subgraph is really significant by itself and not just a product of smaller subtopologies. A limited

version of this idea for size 4 motifs was shown in [15], but here we aim for a generic method (that works for any feasible K) and that is also efficient.

Our main contributions to the stated problem are the following:

- A method that generates random networks using the invariant of subgraphs of frequency $K - 1$, up to a certain margin, with an algorithm based on simulated annealing [10];
- A study of different ways of applying the previous method by using additional invariants like the classic degree sequence invariant;
- An algorithm, based on [17, 25], that updates the frequency of subgraphs after an edge addition or removal, which is used in order to compute the frequencies of subgraphs of size $K - 1$ that the mentioned method requires;

We analyze our method to show that it is both efficient and accurate. To do so, we rely on different real complex networks and show that our method obtains different results when comparing with the classic degree sequence model. We also show that our frequency update algorithm performs much better than recalculating all frequencies in every iteration of the generation method.

The rest of this paper is organized as follows. Section 2 discusses some preliminaries and background concepts regarding network motif analysis, needed for the following sections. Section 3 presents our generation method and also show some of its properties. In section 4 we showcase our frequency updating algorithm and prove its correctness. Section 5 contains a brief experimental analysis of our proposed methods and algorithms. Finally we conclude in Section 6.

1.1 Related Work

Milo et al. [15] use, as null model, random graphs that maintain the degree sequence and subgraph count of size $K - 1$, when calculating motifs of size K . Their implementation uses a Monte Carlo Metropolis-Hastings algorithm for directed networks to calculate motifs of size 4, but does not suggest an immediate strategy for undirected networks or subgraph size greater than 4.

In other related work, Bois and Gayraud in [3] use prior probability to generate random graphs with a given count of subgraphs, but only present priors for two types of directed subgraphs of size 3. Ritchie et al. [21] present an algorithm parametrized by a degree sequence and a set of subgraphs that generates random graphs with those parameters. It is based on the matching algorithm [14], whereas our work uses a Markov chain Monte Carlo method of generation.

We also note that, as far as we know, there is no known method that efficiently updates subgraph frequencies on an edge addition or removal.

2 Network Motif Finding

2.1 Definition of Network Motif

The concept of motifs as building blocks of networks was first described by Milo et al. in [15] as patterns of inter-connections occurring in numbers that are

significantly higher than what one would expect. To simplify notation, we will refer to network motifs simply as motifs.

A determined subgraph is considered significant if its frequency in the original graph is exceptionally high in comparison with its frequency on random networks under a certain null model. To assess exceptionality, one computes the probability that the number of times the subgraph appears on a randomized network is lower than on the original network and then compares it with a certain threshold P . This probability can be estimated using Z-scores on a standard normal distribution, by computing the standardized difference between the observed and expected frequency.

To be classified as a motif, according to the original definition [15], it is also required to fulfill two other properties. For a given subgraph, let f_o be the frequency of the subgraph on the original network and f_r the average frequency of the same subgraph on random networks with an unspecified null model. The first constraint is minimal frequency, that is, f_o has to have a minimum value of U , to ensure a quantitative minimum. The second constraint is minimal deviation, that is, f_o needs to be significantly larger than f_r , to prevent the detection of motifs that have a small difference between these two values but have a narrow distribution in the random networks. This can be stated as $f_o - f_r > D \cdot f_r$, where D is a proportionality threshold.

With this information, we can give a formal definition of motif. Given a set of parameters $\{P, U, D\}$, a subgraph of a given graph is considered a motif if:

- $P(f_r > f_o) \leq P$ (**over-representation**)
- $f_o \geq U$ (**minimal frequency**)
- $f_o - f_r > Df_r$ (**minimum deviation**)

2.2 Algorithms for Subgraph Counting

The main primitive of motif finding is counting subgraphs on graphs, which is called a subgraph census. There are essentially three different ways of doing so: in a network centric way, which corresponds to counting the occurrences of all subgraphs up to a certain size K ; in a subgraph centric way, which corresponds to counting the occurrences of a single subgraph; in a set centric way, which corresponds to counting the occurrences of a set of subgraphs.

The state of the art algorithms that do a generic network centric census are QuateXelero [9] and FaSE [17], which are similar contemporaneous algorithms. Both build on previous methods [25] that do an enumeration of all subgraphs up to a certain size K and then perform isomorphism tests on each one using a tool like `nauty` [11]. By building an intermediate structure (a quaternary tree and a g-trie, respectively) the number of necessary isomorphism tests is decreased to a multiple of the number of different types of subgraph present in the network. More recently, some methods [12, 18] explore combinatorial properties of graphs to achieve algorithms that are orders of magnitude better than any generic method, but that can only work with subgraphs up to a certain size (currently up to 5 for undirected graphs [18] and 4 for directed [12]).

The most well known subgraph centric algorithm is the work by Grochow and Kellis [8], which efficiently counts the frequency of a single subgraph using a set of generated symmetry breaking conditions. Finally, there is only one known set centric algorithm, the work by Ribeiro and Silva [20].

2.3 Random Graphs

The study of random graphs is growing rapidly as a model of complex networks. Although the research on this topic dates back to the late 1950s, where, in a series of publications, Paul Erdos and Alfréd Rényi [6, 5] introduce a model, known as Erdos-Rényi (ER). In this model, each pair of vertices is connected with an independent probability p . More recently, other models have been proposed that follow closely characteristics from real world networks. Among these, Watts and Strogatz [24], propose a model to generate small-world graphs, networks whose average path length grows proportionally to the logarithm of the number of nodes in the network, and Barabasi-Albert [1] introduce another model for scale-free graphs [2], where the degree distribution follows a power law.

When focusing on more local properties, random graphs using a given degree sequence have become one of the most studied models, after their widespread use as null model for network motifs discovery [13, 15]. There is a multitude of algorithms to generate this type of graphs, of which we highlight the main two:

- The switching method [19] uses a Markov chain, starting with an initial network with the desired degree sequence and carries out a series of Monte Carlo switches that preserve that sequence.
- The matching algorithm [16] is based on “stubs”. Each vertex is assigned a set of edge extremities, either incoming or outgoing. For each of these stubs, the vertex tries to connect with another one with the opposite type of stub.

On their original work, Milo et al. [15] use as null model both the degree sequence and subgraph frequency of size 3. To achieve this, they use the switching method to preserve the degree sequence and a Monte Carlo Metropolis-Hastings algorithm to approximate the subgraph count of the referred size. The frequency vectors are updated using analytical expressions using the neighbours of the vertices used for the edge switch.

3 Generation of Random Graphs

In this section, we discuss a generator of random graphs, with the novelty of allowing the random networks to be generated with approximately the same frequency of subgraphs of size $K - 1$ as an original network. We also permit the graphs to maintain or vary their degree sequence. The generation procedure is split in two phases: *randomization* and *convergence*.

3.1 Randomization

We offer three ways of creating an initial network. The first two employ a Markov chain edge swapping technique like in [15] and the third is a classical ER model, with number of edges equal to the number of edges in the original network.

The two Markov chain algorithms we utilize are similar, they both start with a real network and perform edge switches. The first version, which maintains degree sequence, given different nodes A , B , C and D , with connections $A \rightarrow B$ and $C \rightarrow D$, removes these existing connections and adds the new edges, $A \rightarrow D$ and $C \rightarrow B$. Nodes are selected in a way that ensures the prior inexistence of these two new connections. We do not distinguish between single and double edges, considering double edges simply as two independent single ones. The undirected case is easily generalizable.

The second type of Markov chain edge swap modifies the out-degree sequence of the network, for directed networks, and both in and out-degree sequences, in undirected networks. Given different nodes A , B and C , we delete the connection $A \rightarrow B$ and annex the edge $C \rightarrow B$, reducing the out-degree of node A by 1, while incrementing C 's by the same amount. As before, nodes are selected with the requirement that A is connected to B but C is not.

The difference between the initial graphs produced by these two Markov chain variants lies in the time taken to converge to the desired subgraph count, the first version requires a lesser number of iterations. However, both produce graphs with a similar level of *energy*. Given two vectors (V_1 and V_2) with the number of appearances of each type of subgraph, where Γ denotes the set of these subgraphs, in two different networks, we define *energy* as the distance between these two vectors and calculate it as:

$$e = \frac{\sum_{i \in \Gamma} \frac{|V_{1,i} - V_{2,i}|}{V_{1,i} + V_{2,i}}}{|\Gamma|}$$

We refer to the energy of a random network as the distance between its vector of subgraph frequency and the corresponding vector from the original network.

For both Markov chain schemes, we repeat the edge swapping process $\mathcal{O}(E)$ times, where E represents the number of edges in the graph. The constant used is diverse in the existing literature, so we studied how the energy varies in function of the number of switches applied to the original network. We observed that a higher number of switches does not lead to higher energy. It should be noted that energy is not the sole measure of how well a graph is randomized and a low number of switches may not cause enough impact on other measures.

3.2 Convergence

After generating the initial network, we start the process of switching edges to obtain a subgraph count close to that of the real network. The convergence phase stops when the energy reaches a certain tunable threshold, where energy equal to

0 means that the subgraph frequencies of the random network and the original network are the same. In this phase, we use simulated annealing [10].

Simulated annealing is a metaheuristic technique used to approximate the global optimum of a large search space. On a general case, on each iteration, the heuristic chooses a random neighbouring state of the current state and decides probabilistically between changing to the new state or staying in the current one. This process is repeated until a global optimum solution is found or a solution that differs from the optimum less than a given threshold.

In our implementation of the method, the neighbouring state is chosen using the edge swapping mechanism described previously. If our initial network was obtained through the ER model or the out-degree changing Markov chain method, the swap also uses the out-degree changing switch. Otherwise, if the degree sequence was maintained throughout the randomization process, we only perform the type of switch that preserves it.

In order to decide if the the new candidate graph is accepted, we use an acceptance probability function $P(e, e', t)$, where e represents the current graph's energy, e' the candidate graph's energy and t is a parameter that decays over time, called the *temperature*. We use the same acceptance function as in the original formulation by Kirkpatrick et al. in [10], if $e' < e$, we always accept the transition, otherwise, we accept it with probability $\exp(\frac{e-e'}{t})$.

A feature of simulated annealing is the decreasing temperature over time. This forces the state to converge to an optimum as, with lower temperature, the probability of accepting a state with higher energy is lessened. Upon reaching a point in the computation where the temperature reaches 0, only states with lesser energy are accepted and the computation eventually stops. The rate at which the temperature decreases is called the cooling factor of the algorithm.

4 Updating frequencies of subgraphs

The main bottleneck of the method described in the previous section is computing the frequencies of subgraphs in every iteration, to estimate the energy of the current solution. In [15], an analogous operation was done recounting the frequencies of subgraphs after each iteration of their algorithm until convergence. Our approach avoids recomputing all of the frequencies by only considering the subgraphs that are changed by the addition or removal of a certain edge.

The base of our method is the FaSE [17] algorithm, which we will extend in order to only count subgraphs that touch a given edge. Firstly, we will briefly describe the algorithm.

4.1 FaSE Algorithm

The original FaSE algorithm enumerates all connected subgraphs of a given size K and in the end computes the isomorphism of some of the subgraphs. To avoid having to compute the isomorphism of all subgraphs, the algorithm partitions subgraphs into intermediate classes during the enumeration process. By requiring

that all subgraphs in one of the intermediate classes are isomorphic, in the end we only need to compute one isomorphism test per class. This is done by encapsulating the topological features of the enumerating graph in a tree like data structure. Thus, we can divide the algorithm into two interleaved concepts: the enumeration and a tree data structure.

Enumeration: The enumeration step can be done using any algorithm that grows a set of connected vertices. The algorithm from [25], ESU, was chosen since it is simple, efficient and fulfills all the requirements. We will describe its functioning since it will be useful for the end of this section.

ESU works by enumerating all size K subgraphs exactly once. It does so by keeping two ordered sets of vertices: V_s , which represents the partial subgraph that is currently being enumerated; V_{ext} , which represents the set of vertices that can be added to V_s as a valid extension. Each vertex is represented by a label which is unique and defined between 1 and $|V|$.

For each vertex v the algorithm repeats the same procedure setting initially $V_s = \{v\}$ and $V_{ext} = N(v)$, where $N(v)$ are the neighbors of v . This procedure starts by removing one element u of V_{ext} at a time. For each u , a new V'_s and V'_{ext} are created and the same procedure is repeated. V'_s is set to $V_s \cup \{u\}$ and V'_{ext} is set to V_{ext} without u and with additionally each element in $N_{exc}(u, V_s)$ with value greater than v . $N_{exc}(u, V_s)$ are the exclusive neighbors of u given V_s , that is, the neighbors of u that are not neighbors of elements in V_s . This procedure stops when the size of V_s reaches K , in which case V_s contains one occurrence of size K . The addition of elements in $N_{exc}(u, V_s)$ along with the $u > v$, ensure that there is no subgraph enumerated twice, and it can be proved [25] that this procedure stops and enumerates all subgraphs.

The tree data structure: During the enumeration process, this data structure is used to encapsulate information about the subgraph contained in V_s . Since this is a recursive procedure, one can use information about the initial content of V_s to build a partial isomorphism representation, that can be complemented on each vertex insertion in V_s . For this, a data structure called a gtrie is used, which is similar to a prefix tree of subgraphs. Whenever a new vertex is added to V_s , one uses the information of connectivity with the previous elements of V_s to generate a label that identifies the current partial subgraph, which is used as the identifier for the mentioned intermediate classes.

Figure 1 summarizes the whole algorithm. The tree on the left represents the implicit recursion tree ESU creates. The induced g-trie on the right is a visual representation of the actual g-trie FaSE creates. More information about the FaSE can be found in [17].

4.2 FaSE with updates

Our method to efficiently update frequency counts works by altering the enumeration algorithm to count frequencies starting on edges. When adding an edge, the algorithm first counts all subgraphs that use the edge's two ends and decrements their frequency. Afterwards, it adds the new edge and counts all subgraphs that

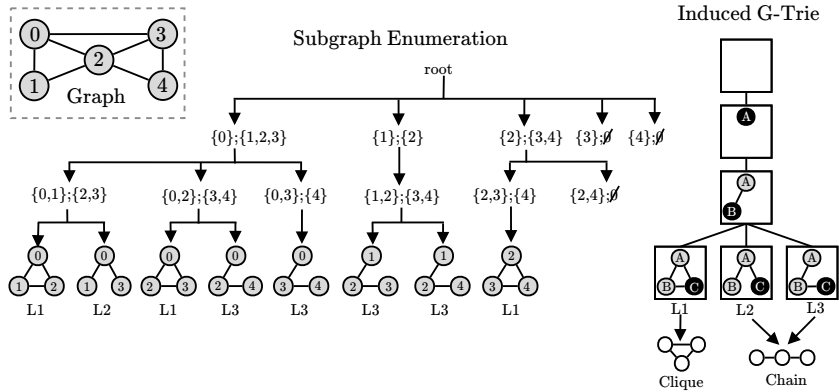


Fig. 1. Summary of the FaSE algorithm.

touch that edge. To remove an edge we do an analogous process. Our method is based on the ESU algorithm, altering it to start on a given edge.

For a given edge to add, $\{a, b\}$, the algorithm first considers as initial sets $V_s = \{a, b\}$ and $V_{ext} = N(a) \cup N(b) \setminus \{a, b\}$ and only uses these as initial sets (meaning it does not recurse on other initial V_s and V_{ext}). The rest of the procedure is similar to the original ESU algorithm, but the symmetry breaking is removed, that is, when adding a node u' to V_{ext} , there is no comparison with a : if u' belongs to $N_{exc}(u, V_s)$ it will be added to V_{ext} .

To prove that this method is correct we use the original correction proof of the ESU algorithm. If a is the minimal node of the graph (that is, for every node v , $a \leq v$), all subgraphs that include a will be enumerated on the first iteration of the algorithm. For that iteration, if b is the first element of N_{ext} , then it will be removed and the next iteration has $V_s = \{a, b\}$ and $V_{ext} = N(a) \setminus \{b\} \cup N_{exc}(b, \{a\}) = N(a) \cup N(b) \setminus \{a, b\}$. Since this is the only recursion path that will include a and b (since b was the first node to be removed from the initial N_{ext}), all subgraphs that contain a and b will be counted on this recursive subtree. Since this is analogous to our method, its correctness implies the correctness of our method.

5 Experimental Evaluation

We apply our techniques to four networks, two of them neurobiological, based on [23]. The neurobiological networks are directed and represent a macaque visual cortex, with 30 nodes and 311 connections, and a macaque cortex, with 71 nodes and 746 edges. The other two networks are undirected and represent a social network of jazz musicians [7], with 198 nodes and 2742 edges, and a geo-spatial network of a power grid in the United States [24], with 4941 nodes and 6594 edges.

We measure the significance of subgraphs of size $K = 4$ and $K = 5$, using the Z-score metric. For each network and each type of initial random network,

we generate an ensemble of 100 random networks. For the convergence phase, we define our energy threshold as 5%, if the vectors of subgraphs count differ in 5% or less, we stop the computation and output the network as it is at that point. We use an initial temperature of 0.01 and a cooling factor of 0.99. Table 1 presents results for the mentioned networks, by comparing the Z-score calculated by our methods against simply maintaining the degree sequence.













Network	K	Subgraph	Original	Keep $K - 1$		
				Keep Deg. Seq.	Change Deg. Seq.	ER
Macaque Cortex	4		61.20 ^a	-2.29	-0.71	-4.41
			182.30 ^a	6.19	2.47	12.66
			-10.17 ^b	12.01	10.64	15.20
Macaque Visual Cortex	4		36.76 ^a	-1.58	-0.63	-2.88
			14.63 ^a	-2.29	-2.20	-2.61
			-3.49 ^b	12.01	4.90	5.40
	5		278.57 ^b	4.11	3.85	-0.71
		117.72 ^b	8.79	6.41	1.62	
Power	5		82.83 ^b	4.88	-3.45	2.86
			-21.57 ^b	-18.25	-17.65	0.09
Jazz	5		438.35 ^b	60.47	29.62	15.82
			-45.84 ^b	-17.31	6.18	70.54

Table 1. Z-score results for some subgraphs in the macaque cortex and macaque visual cortex networks. ^a result was taken from [23]. ^b was calculated by us, using degree sequence invariance as null model.

Using our generator as null model, the Z-score of the first and second subgraphs on the macaque cortex and fourth, fifth, seventh and eighth on the

macaque visual cortex was significantly lower than the Z-score calculated using solely the degree sequence as invariant. We speculate that these subgraphs, which are considered over-represented in the original network by Sporns et al. [23], are simply a consequence of the prevalence of their induced subgraphs of size $K - 1$. By preserving the frequency of the latter, the former become more common in the generated random networks.

On the other hand, subgraphs third and sixth from macaque cortex and macaque visual cortex respectively, are originally considered under-represented but, under our generator, can be considered motifs. Note that the Z-score values are similar using different initial perturbations on the original networks.

On the **power** network, we show a subgraph of size 5 that was considered a motif under the previous model, but with our new model, it is not considered over-represented anymore. The other example for the same network, using a Markov chain edge swap as the initial network yields a similar Z-score as the original model, but converging from an ER network produces a significantly different score.

For the **jazz** network, we present an example where an extremely over-represented subgraph is still considered a motif under our model. It is the size 5 clique and its over-representation can not be simply explained by the number of size 4 cliques. In the other example, each of the models for the initial random network provides a substantially different Z-score, from being considered under-represented if the Markov chain edge swap process that retains the degree sequence is used, to being treated as motif if the initial network follows the ER model.

We also study the improvement obtained by efficiently updating subgraph counts. To this end, Table 2 shows the average execution time, in seconds, for each network, comparing the efficient update against running a full census after each edge swap. These tests were run with initial temperature 0.01, cooling factor set to 0.99 and using the Markov chain edge swap variant that preserves the degree sequence. Subgraph frequency of size 3 was maintained for the macaque networks and size 4 for the **power** and **jazz** networks.

	Macaque Cortex	Macaque Visual Cortex	Power	Jazz
Efficient Update (s)	64.85	0.22	239.56	1034.06
Full Census (s)	103.58	12.35	4274.47	25102.0
Speedup (\times faster)	1.6	56.1	17.8	24.3

Table 2. Average execution time, in seconds, and speedup, of the efficient update in comparison with the full census, to generate a random network preserving the frequency of subgraphs of size 3 for the neurobiological networks and size 4 for the **jazz** and **power** networks.

For the macaque cortex network, in average, each network took nearly twice as much doing the full census after each edge switch than using our efficient

frequency update. However, for the **jazz** and **power** networks, in average, each network was 1 order of magnitude faster using the efficient update technique and the macaque visual cortex was about 2 orders of magnitude faster.

Clearly, both macaque networks are outliers of efficiency, probably because they are both small dense networks. Our efficient update method works best for larger sparse networks, because in this case, on average, the number of subgraphs that change after a single edge addition or removal is only a small fraction of the total number of subgraphs. In this sense, the **jazz** and **power** networks are better fits for this model, as are most social networks.

6 Conclusion

We introduced a generator of random graphs that preserves the frequency of subgraphs of size $K - 1$. The generation is split in two phases, where the original networks first suffers an initial perturbation, via a Markov chain edge swapping technique or a classic Erdos-Renyi model, and then converges to the desired frequency up to a difference of percentage threshold, using simulated annealing.

We applied our generator to four real complex networks and compared the significance of different subgraphs against results published in [23]. The Z-score calculated by using our generator as null model is significantly lower for certain subgraphs of size K , which can be explained by the prevalence of induced subgraphs of size $K - 1$.

We also devised a technique to efficiently update the frequency of subgraphs after an addition or removal of a single edge. In summary, it works by searching all the subgraphs that touch the edge's endpoints and updates their frequency. This technique is critical to the convergence phase of our generator, as it is, on average, at least 2 times faster and in many cases orders of magnitude faster than running the full networks census from scratch.

Acknowledgements: This work is funded within FourEyes, a research line within project *TEC4Growth/NORTE-01-0145-FEDER-000020*.

References

1. Albert, R., Barabasi, A.L.: Statistical mechanics of complex networks. *Reviews of Modern Physics* 74, 47–97 (2002)
2. Barabási, A.L., Albert, R.: Emergence of scaling in random networks. *Science* 286(5439), 509–512 (1999)
3. Bois, F.Y., Gayraud, G.: Probabilistic generation of random networks taking into account information on motifs occurrence. *Journal of Computational Biology* 22(1), 25–36 (2015)
4. Choobdar, S., Ribeiro, P., Bugla, S., Silva, F.: Comparison of co-authorship networks across scientific fields using motifs. In: *ASONAM, 2012 IEEE/ACM International Conference on*. pp. 147–152. IEEE (2012)
5. Erdos, P., Rényi, A.: On the evolution of random graphs. *Bull. Inst. Internat. Statist* 38(4), 343–347 (1961)
6. Erdos, P., Rényi, A.: On random graphs i. *Publ. Math. Debrecen* 6, 290–297 (1959)

7. Gleiser, P.M., Danon, L.: Community structure in jazz. *Advances in complex systems* 6(04), 565–573 (2003)
8. Grochow, J.A., Kellis, M.: Network motif discovery using subgraph enumeration and symmetry-breaking. In: *Annual International Conference on Research in Computational Molecular Biology*. pp. 92–106. Springer (2007)
9. Khakabimamaghani, S., Sharafuddin, I., Dichter, N., Koch, I., Masoudi-Nejad, A.: Quatexelero: an accelerated exact network motif detection algorithm. *PloS one* 8(7), e68073 (2013)
10. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. *Science* 220(4598), 671–680 (1983)
11. McKay, B.D., Piperno, A.: Practical graph isomorphism, ii. *Journal of Symbolic Computation* 60, 94–112 (2014)
12. Meira, L.A., Maximo, V.R., Fazenda, A.L., da Conceicao, A.F.: Accelerated motif detection using combinatorial techniques. In: *SITIS, 2012 Eighth International Conference on*. pp. 744–753. IEEE (2012)
13. Milo, R., Itzkovitz, S., Kashtan, N., Levitt, R., Shen-Orr, S., Ayzenshtat, I., Sheffer, M., Alon, U.: Superfamilies of evolved and designed networks. *Science* 303(5663), 1538–1542 (2004)
14. Milo, R., Kashtan, N., Itzkovitz, S., Newman, M.E., Alon, U.: On the uniform generation of random graphs with prescribed degree sequences. *arXiv preprint cond-mat/0312028* (2003)
15. Milo, R., Shen-Orr, S., Itzkovitz, S., Kashtan, N., Chklovskii, D., Alon, U.: Network motifs: simple building blocks of complex networks. *Science* 298(5594), 824–827 (2002)
16. Molloy, M., Reed, B.: A critical point for random graphs with a given degree sequence. *Random structures & algorithms* 6(2-3), 161–180 (1995)
17. Paredes, P., Ribeiro, P.: Towards a faster network-centric subgraph census. In: *ASONAM, IEEE/ACM International Conference on*. pp. 264–271. IEEE (2013)
18. Pinar, A., Seshadhri, C., Vishal, V.: Escape: Efficiently counting all 5-vertex subgraphs. *arXiv preprint arXiv:1610.09411* (2016)
19. Rao, A.R., Jana, R., Bandyopadhyay, S.: A markov chain monte carlo method for generating random $(0, 1)$ -matrices with given marginals. *Sankhyā: The Indian Journal of Statistics, Series A* pp. 225–242 (1996)
20. Ribeiro, P., Silva, F.: G-tries: a data structure for storing and finding subgraphs. *Data Mining and Knowledge Discovery* 28(2), 337–377 (2014)
21. Ritchie, M., Berthouze, L., Kiss, I.Z.: Generation and analysis of networks with a prescribed degree sequence and subgraph family: higher-order structure matters. *Journal of Complex Networks* p. cnw011 (2016)
22. Shen-Orr, S.S., Milo, R., Mangan, S., Alon, U.: Network motifs in the transcriptional regulation network of *escherichia coli*. *Nature genetics* 31(1), 64–68 (2002)
23. Sporns, O., Kötter, R.: Motifs in brain networks. *PLoS Biol* 2(11), e369 (2004)
24. Watts, D., Strogatz, S.: Collective dynamics of small-world networks. *Nature* 393, 440–442 (1998)
25. Wernicke, S.: Efficient detection of network motifs. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)* 3(4), 347–359 (2006)