On the uniform generation of random graphs with prescribed degree sequences

R. Milo,[1,2] N. Kashtan,[2,3] S. Itzkovitz,[1,2] M. E. J. Newman,[4] and U. Alon[1,2]

[1]*Department of Physics of Complex Systems, Weizmann Institute of Science, Rehovot, Israel 76100*
[2]*Department of Molecular Cell Biology, Weizmann Institute of Science, Rehovot, Israel 76100*
[3]*Department of Computer Science and Applied Mathematics,*
Weizmann Institute of Science, Rehovot, Israel 76100
[4]*Department of Physics and Center for the Study of Complex Systems,*
University of Michigan, Ann Arbor, MI 48109–1120, U.S.A.

Random graphs with prescribed degree sequences have been widely used as a model of complex networks. Comparing an observed network to an ensemble of such graphs allows one to detect deviations from randomness in network properties. Here we briefly review two existing methods for the generation of random graphs with arbitrary degree sequences, which we call the "switching" and "matching" methods, and present a new method based on the "go with the winners" Monte Carlo method. The matching method may suffer from nonuniform sampling, while the switching method has no general theoretical bound on its mixing time. The "go with the winners" method has neither of these drawbacks, but is slow. It can however be used to evaluate the reliability of the other two methods and, by doing this, we demonstrate that the deviations of the switching and matching algorithms under realistic conditions are small compared to the "go with the winners" algorithm. Because of its combination of speed and accuracy we recommend the use of the switching method for most calculations.

I. INTRODUCTION

In the rapidly growing literature on the modeling of complex networks one of the most important classes of network models is the random graph [1]. One well-studied such model is the model consisting of the ensemble of all graphs that have a given degree sequence [2, 3, 4, 5, 6], and this model has proved useful in understanding a variety of network properties. Realistic applications often require that we restrict ourselves to graphs with no multiple edges between any vertex pair and no self-edges. Unfortunately, both the analytic and numerical study of such networks is known to present challenges [2, 7, 8, 9, 10, 11, 12, 13, 14]. In this short paper we consider computer algorithms for generating graphs uniformly from this ensemble. We are concerned primarily with directed graphs, since the examples we will consider are directed, but the concepts discussed generalize in a straightforward fashion to the undirected case also.

There are two algorithms in common use for the generation of random graphs with single edges. We will refer to them as the *switching algorithm* [8, 9, 15, 16, 17, 18, 19] and the *matching algorithm* [4, 5, 19]. We argue that, under certain circumstances, both of these algorithms can generate a nonuniform sample of possible graphs. We then present a new algorithm based on the Monte Carlo procedure known as *go with the winners* [22, 23], which generates uniformly sampled graphs. We compare the three methods in the context of a particular network problem—estimation of the density of commonly occurring subgraphs or *motifs*—and show that, in this context, the difference between them is small. This result is of some practical importance, since the "go with the winners" algorithm, although statistically correct, is slow, while the other two algorithms are substantially faster.

II. ALGORITHMS

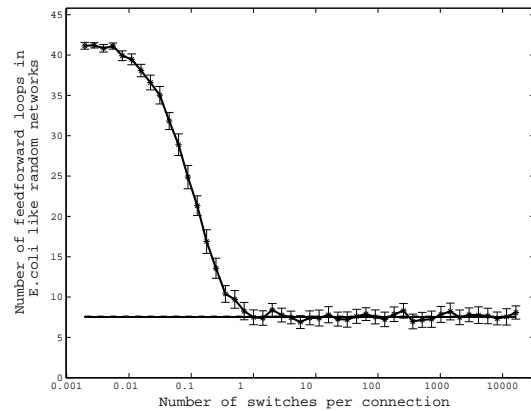In this section we describe the three algorithms under consideration.

FIG. 1: Starting with the transcription network of E. coli, the network is randomized using the switching algorithm described in the text. We plot the number of feed-forward loops in the randomized networks vs. number of switches performed per edge in the graph. The dashed line is the expected asymptotic value obtained using the "go with the winners" algorithm. Each point is an average over one hundred repetitions of the calculation. Error bars are ±3 standard deviations. The randomized network reaches the equilibrium value around one switch per edge on average. Similar results are obtained for other networks and other motifs.

Network	E. coli transcription			yeast transcription			C. elegans neurons			electronic circuit S15850 [24]		
	mean	s.d.	Z	mean	s.d.	Z	mean	s.d.	Z	mean	s.d.	Z
"go with the winners"	7.57(5)	3.05(3)	10.6(1)	11.06(6)	3.60(4)	14.1(2)	88(1)	10.7(7)	3.4(3)	2.20(5)	1.48(3)	284(6)
switching	7.63(9)	3.05(6)	10.5(2)	11.0(1)	3.71(7)	13.7(3)	88.3(3)	10.1(2)	3.6(1)	2.24(5)	1.47(3)	286(6)
matching	7.67(9)	2.98(6)	10.8(2)	11.1(1)	3.67(7)	13.8(3)	94.5(3)	10.0(2)	3.0(1)	2.21(5)	1.45(3)	290(6)

TABLE I: Mean and standard deviation (s.d.) of the number of appearances of the feed-forward loop subgraph in random networks with degree sequences the same as the real world networks studied in [19]. We used between 1000 and 10 000 random networks for each measurement. Z-scores are the number of standard deviations by which the real network deviates from the average of the random ensemble. Standard errors are shown in parentheses.

A. Switching algorithm

First, we describe the switching algorithm, which uses a Markov chain to generate a random graph with a given degree sequence [8, 9, 15, 16, 17, 18, 19]. For simplicity, we discuss directed networks with no mutual edges (vertex pairs with edges running in both directions between them). The case with mutual edges is a simple generalization [9].

The method starts from a given network and involves carrying out a series of Monte Carlo switching steps whereby a pair of edges $(A \to B, C \to D)$ is selected at random and the ends are exchanged to give $(A \to D, C \to B)$. However, the exchange is only performed if it generates no multiple edges or self-edges; otherwise it is not performed. The entire process is repeated some number QE times, where E is the number of edges in the graph and Q is chosen large enough that the Markov chain shows good mixing. (Exchanges that are not performed because they would generate multiple or self-edges are still counted to insure detailed balance [25].)

This algorithm works well but, as with many Markov chain methods, suffers because in general we have no measure of how long we need to wait for it to mix properly. Theoretical bounds on the mixing time exist only for specific near-regular degree sequences [10]. We empirically find, however, that for many networks, values of around $Q = 100$ appear to be more than adequate (see Fig. 1).

B. Matching algorithm

An alternative approach is the matching algorithm [4, 5, 19], in which each vertex is assigned a set of "stubs" or "spokes"—the sawn-off ends of incoming and outgoing edges—according to the desired degree sequence. (One can also assign mutual-edge stubs for networks that include such edges.) Then in-stubs and out-stubs are picked randomly in pairs and joined up to create the network edges. If a multiple or self-edge is created, the entire network is discarded and the process starts over from scratch.

This process will correctly generate random directed graphs with the desired properties. Unfortunately, however, many real-world networks have a heavy-tailed degree distribution that includes a small minority of vertices with high degree. All other things being equal, the expected number of edges between two such vertices will often exceed one, making it unlikely that the procedure above will run to completion, except in the rarest of cases. To obviate this problem a modification of the method can be used in which, following selection of a stub pair that creates a multiple edge, the network is not discarded, and an alternative stub pair is selected at random. In general this method generates a biased sample of possible networks [21] but, as we will show, not significantly so for our purposes (see Table I).

C. Go with the Winners algorithm

The "go with the winners" algorithm is a non-Markov-chain Monte Carlo method for sampling uniformly from a given distribution [22, 23]. When applied to the problem of graph generation, the method is as follows. We consider a colony of M graphs. As with the matching algorithm, we start with the appropriate number of in-stubs and out-stubs for each vertex and repeatedly choose at random one in-stub and one out-stub from the graph and link them together to create an edge. If a multiple edge or self-edge is generated, the network containing it is removed from the colony and discarded. To compensate for the resulting slow decline in the size of the colony, its size is periodically doubled by cloning each of the surviving graphs; this cloning step is carried out at a predetermined rate chosen to keep the size of the colony roughly constant on average. The process is repeated until all stubs have been linked, then one network is chosen at random from the colony and assigned a weight:

$$W_i = 2^{-c} \frac{m}{M}, \tag{1}$$

where c is the number of cloning steps made and m is the number of surviving networks. The mean of any quantity X (for example, the number of occurrences of a given subgraph) over a set of such networks is then given by

$$\frac{\sum_i W_i X_i}{\sum_i W_i}, \tag{2}$$

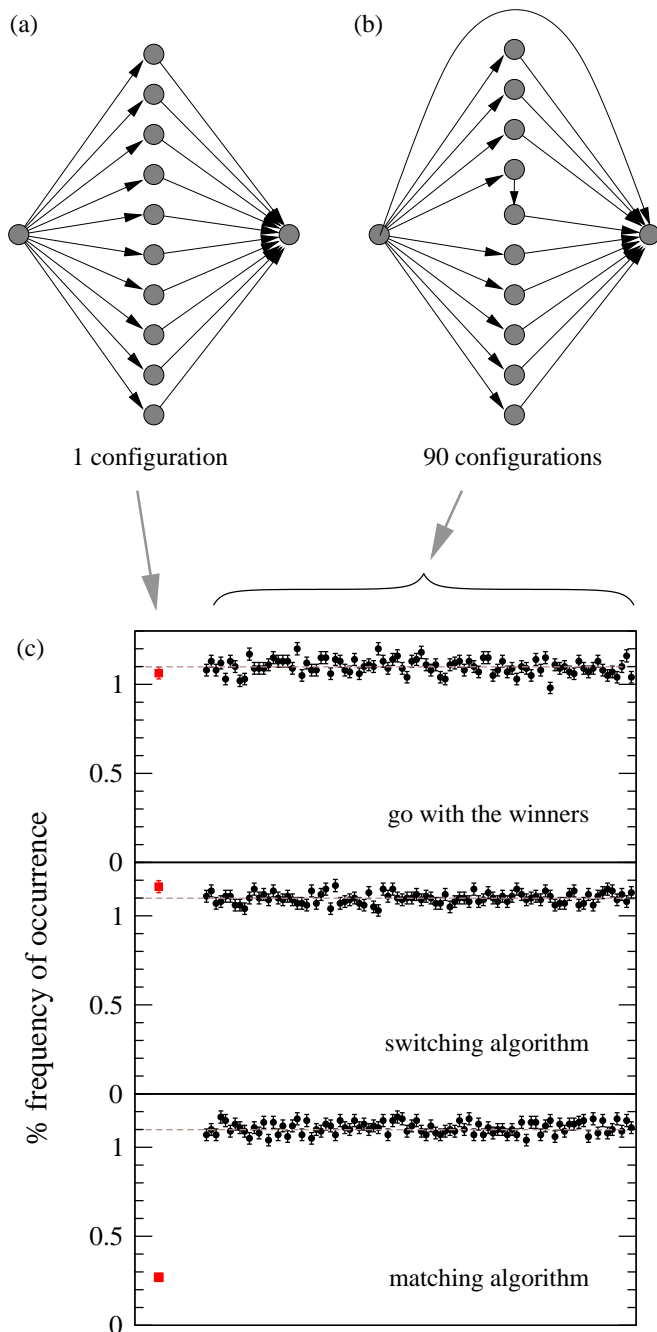where X_i is the value of X in network i.

FIG. 2: Uniformity tests of the three algorithms on a toy network. Panels (a) and (b) depict the two types of topologies of the 91 random networks studied, one of them like (a) and 90 like (b). Panel (c) shows the frequency with which each configuration is sampled by our three algorithms. 100 000 graphs were generated with each algorithm, and the figure shows the fraction of graphs of each type generated. If sampling were uniform, each should appear with probability $\frac{1}{91}$, which is indicated by the dotted lines. The "go with the winners" and switching algorithms sample uniformly within sampling error, passing both the Kolmogorov–Smirnoff and Lillie Gaussian tests. The matching algorithm under-samples the unique configuration (a).

III. COMPARISON OF ALGORITHMS

In Fig. 2 we show a comparison of the performance of our three algorithms when applied to a simple toy network. The network consists of an out-hub with ten outgoing edges, an in-hub with ten incoming edges, and ten nodes with one incoming edge and one outgoing edge each. Given this degree sequence, there are just two distinct network topologies with no multiple edges, as shown in Fig. 2a and 2b. There is only a single way to form the network in 2a, but there are 90 different ways to form 2b.

We generated 100 000 random networks using each of the 3 methods described here and the results are summarized in Fig. 2c. As the figure shows, the matching algorithm introduces a bias, undersampling the configuration of Fig. 2a. This is a result of the dynamics of the algorithm, which favors the creation of edges between hubs. The switching and "go with the winners" algorithms on the other hand sample the configurations uniformly, generating each graph an equal number of times within the measurement error on our calculations. The "go with the winners" algorithm truly samples the ensemble uniformly but is far less efficient than the two other methods. The results given here indicate that the switching algorithm produces essentially identical results while being a good deal faster. The matching algorithm is faster still but samples in a measurably biased way.

Now consider the study of network motifs. We are interested in knowing when particular subgraphs or motifs appear significantly more or less often in a real-world network than would be expected on the basis of chance, and we can answer this question by comparing motif counts to random graphs. Some results for the case of the "feed-forward loop" motif [18, 19] are given in Table I. In this case the densities of motifs in the real-world networks are many standard deviations away from random, which suggests that any of the present algorithms is adequate for generating suitable random graphs to act as a null model, although the "go with the winners" and switching algorithms, while slower, are clearly more satisfactory theoretically. The matching algorithm was measurably nonuniform for our toy example above, but seems to give better results on the real-world problem.

Overall, our results appear to argue in favor of using the switching method, with the "go with the winners" method finding limited use as a check on the accuracy of sampling. Accuracy checks are also supplied by analytical estimates for subgraph numbers [12]. Numerical results in [12, 19, 20] were done using the switching algorithm.

IV. CONCLUSIONS

In this paper we have compared three algorithms for generating random graphs with prescribed degree sequences and no multiple edges or self-edges. Two of the three have been used previously, but suffer from nonuni-

formity in their sampling properties, while the third, a method based on the "go with the winners" Monte Carlo procedure, is new and samples uniformly but is quite slow. Of the two older algorithms, we show that one, which we call the "matching" algorithm, has measurable deviations from uniformity when compared to the "go with the winners" method, although for graphs typical of practical studies these deviations are small enough to make no significant difference to most previously published results. The other older algorithm, which we call the "switching" algorithm and which is based on a Markov chain Monte Carlo method, samples correctly in the limit of long times and in practice is found to give good results when compared with the "go with the winners" method. Overall, therefore, we conclude that the switching algorithm is probably the algorithm of choice, with the "go with the winners" algorithm finding a supporting role as a check on uniformity, although its slowness makes it impractical for large-scale use.

[1] B. Bollobas, *Random Graphs*, 2nd edition, Academic Press, New York (2001).

[2] E. Bender and E. Canfield, The asymptotic number of labelled graphs with given degree sequences, *J. Combin. Theory Ser. A* **24**, 296–307 (1978).

[3] M. Molloy and B. Reed, The size of the giant component of a random graph with a given degree sequence, *Combinatorics, Probability and Computing* **7**, 295–305 (1998).

[4] M. Molloy and B. Reed, A critical point for random graphs with a given degree sequence, *Random Structures and Algorithms* **6**, 161–179 (1995).

[5] M. E. J. Newman, S. H. Strogatz, and D. J. Watts, Random graphs with arbitrary degree distribution and their applications, *Phys. Rev. E* **64**, 026118 (2001).

[6] F. Chung and L. Lu, The average distances in random graphs with given expected degrees, *Proc. Natl. Acad. Sci. U.S.A.* **99**, 15879–15882 (2002).

[7] T. A. B. Snijders, Enumeration and simulation methods for 0–1 matrices with given marginals, *Psychometrika* **56**, 397–417 (1991).

[8] A. R. Rao, R. Jana, and S. Bandyopadhya, A Markov chain Monte Carlo method for generating random $(0, 1)$-matrices with given marginals, *Indian J. of Statistics* **58**, 225–242 (1996).

[9] J. M. Roberts, Jr., Simple methods for simulating sociomatrices with given marginal totals, *Social Networks* **22**, 273–283 (2000).

[10] R. Kannan, P. Tetali, and S. Vempala, Simple Markov-chain algorithms for generating bipartite graphs and tournaments, *Proceedings of the ACM Symposium on Discrete Algorithms* (1997).

[11] Y. Chen, P. Diaconis, S. P. Holmes, and J. S. Liu, Sequential Monte Carlo Methods for Statistical Analysis of Tables, *Discussion Paper 03-22*, Institute of Statistics and Decision Sciences, Duke University (2003).

[12] S. Itzkovitz, R. Milo, N. Kashtan, G. Ziv, and U. Alon, Subgraphs in random networks, *Phys. Rev. E* **68**, 026127 (2003).

[13] S. Maslov, K. Sneppen, and A. Zaliznyak, Pattern detection in complex networks: Correlation profile of the Internet, preprint cond-mat/0205379 (2002).

[14] J. Park and M.E.J. Newman, The origin of degree correlations in the Internet and other networks, *Phys. Rev. E* **68**, 026112 (2003).

[15] M. E. J. Newman, Assortative mixing in networks, *Phys. Rev. Lett.* **89**, 208701 (2002).

[16] S. Maslov and K. Sneppen, Specificity and stability in topology of protein networks, *Science* **296**, 910–913 (2002).

[17] A. Roberts, L. Stone, Island-sharing by archipelago species. *Oecologia* **83**, 560-567 (1990).

[18] S. Shen-Orr, R. Milo, S. Mangan, and U. Alon, Network motifs in the transcriptional regulation network of Escherichia coli, *Nature Genetics* **31**, 64–68 (2002).

[19] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan., D. Chklovskii, and U. Alon, Network motifs: Simple building blocks of complex networks, *Science* **298**, 824–827 (2002).

[20] R. Milo, S. Itzkovitz, N. Kashtan, R. Levitt, S. Shen-Orr, I. Ayzenshtat, M. Sheffer and U. Alon, Superfamilies of designed and evolved networks, *Science* **303**, 1538–42 (2004).

[21] O. D. King, Private communication.

[22] D. Aldous and U.V. Vazirani, "Go With the Winners" algorithms, *Proceedings of the IEEE Symposium on Foundations of Computer Science*, pp. 492–501 (1994).

[23] P. Grassberger and W. Nadler, "Go-with-the-winners" simulations, preprint cond-mat/0010265 (2000).

[24] F. Brglez, D. Bryan, and K. Kozminski, Combinatorial Profiles of Sequential Benchmark Circuits, *Proceedings of IEEE International Symposium on Circuits and Systems*, 1929 (1989).

[25] There are singular networks where the Markov process is not ergodic. This lack of ergodicity can be removed by making small modifications as in [8], and by choosing the number of switching steps per edge to be randomly distributed around Q.