Investigating Fitness Measures for the Automatic Construction of Graph Models

Kyle Robert Harrison[1], Mario Ventresca[2(✉)], and Beatrice M. Ombuki-Berman[1]

[1] Department of Computer Science, Brock University, St. Catharines, ON, Canada
{kh08uh,bombuki}@brocku.ca

[2] School of Industrial Engineering, Purdue University, West Lafayette, IN, USA
mventresca@purdue.edu

Abstract. Graph models are often constructed as a tool to better understand the growth dynamics of complex networks. Traditionally, graph models have been constructed through a very time consuming and difficult manual process. Recently, there have been various methods proposed to alleviate the manual efforts required when constructing these models, using statistical and evolutionary strategies. A major difficulty associated with automated approaches lies in the evaluation of candidate models. To address this difficulty, this paper examines a number of well-known network properties using a proposed meta-analysis procedure. The meta-analysis demonstrated how these network measures interacted when used together as classifiers to determine network, and thus model, (dis)similarity. The analytical results formed the basis of a fitness evaluation scheme used in a genetic programming (GP) system to automatically construct graph models for complex networks. The GP-based automatic inference system was used to reproduce two well-known graph models, the results of which indicated that the evolved models exemplified striking similarity when compared to their respective targets on a number of structural network properties.

Keywords: Complex networks · Graph models · Centrality measures · Meta-analysis · Genetic programming

1 Introduction

A *complex network* is a collection of related elements in which the emergent patterns of connections hold significant meaning [23]. Complex networks are referred to as *complex* due to their intricate, tightly coupled structure and semantics, not their size alone [23], and arise in a wide variety of natural and artificial contexts. Examples of natural networks include social networks, which emerge from human

A.M. Mora and G. Squillero (Eds.): EvoApplications 2015, LNCS 9028, pp. 189–200, 2015.
DOI: 10.1007/978-3-319-16549-3_16

interaction, and biological networks, which aim to describe biological processes such as protein-protein interaction [5] and neural networks [24]. An example of artificial networks are technological networks, which describe artificially constructed systems such as the Internet and power-grid networks [1].

Devising algorithms that explain the growth patterns of networks has been a topic of interest for over 50 years [10]. These algorithms, known as *graph models*, are capable of generating networks of arbitrary size which replicate certain statistical and structural properties, such as proportions of transitive connections and path lengths. Graph models have a tremendous number of applications across many domains and allow both interpolation of previous network states and extrapolation of future states; see [16] for an overview of the many uses for graph models. While accurate graph models have many benefits, their applicability is governed by how easily they can be created and/or tailored for a specific network at hand. However, the task of constructing graph models from scratch has traditionally been done manually – a time-consuming and difficult process [23].

Automated approaches to graph model construction have the potential to significantly reduce the time and effort for their construction, especially in scenarios where the networks are large. A number of statistical methods to generate meta-models have been proposed to alleviate the manual effort required in building a graph model [8,17]. Similarly, genetic programming (GP) has also been recently proposed for the automated inference of graph models [2,3,22]. While the statistical methods are limited in the types of networks they can produce (e.g., Kronecker graphs produce log-normal distributions), GP provides a potential solution with less limitations. Furthermore, GP has the potential to reveal the underlying mechanisms that define the connections, whereas previous techniques do not. Although automated approaches alleviate prohibitive factors in graph model construction, they are by no means without their own difficulties. For example, verifying that a model accurately describes the target network is no trivial task, as the concept of similarity is loosely defined and dependent upon the network semantics. Furthermore, the process which created the target network is unknown, requiring the evaluation of the candidate graph model to be done through graph comparisons. It should be noted that graph comparison, in this context, is not an isomorphism problem, as the goal is not to reproduce the target network itself, but rather to infer a model which reasonably approximates the growth patterns that created the target network and, by extension, replicates its structural properties.

This paper first provides an analysis of network centrality measures of which the results are used as a basis for the fitness evaluation of a GP system to automate the construction of graph models. The remainder of this paper is structured as follows. Section 2 introduces the topic of graph models and network measures. Section 3 proposes a meta-analysis framework for the analysis of centrality measures. Section 4 describes the GP system used to automatically infer graph models, while the results of automatic inference are presented in Sect. 5. Finally, concluding remarks are given in Sect. 6.

2 Background

This section briefly introduces the network centrality measures and graph models used throughout the remainder of this study. For brevity, only limited, relevant information is provided.

2.1 Network Centrality Measures

Global network properties, such as the average geodesic path length which provides a sense of the information propagation time, are useful to quantify the overall structure of a network but are limited in that they generally disregard the emergent local behaviors of individual vertices. Fan *et al.* [11] point out that using only topological characteristics to evaluate complex network models can be misleading. As such, this study makes use of more localized, vertex-level properties to assess network similarity. *Centrality* refers to how central or "important" a vertex is within a network. The importance of a vertex is, however, subjectively based on the perception of "importance". As such, many definitions of importance, and corresponding measures of centrality, have been proposed. This work examined six well-known centrality measures, namely degree distribution (D), betweenness (B) [13], closeness (C) [14], local transitivity (LCC)[1], eigenvector centrality (EC) [6], and PageRank (PR) [7].

2.2 Graph Models

Graph models are typically stochastic or probabilistic algorithms which, through repeated execution, produce a set of graphs that depict commonalities with respect to certain properties, but are otherwise random [23]. The common properties among graphs generated by a model are dependent upon the model, but may include basic structural properties such as the degree distribution [4] and path lengths [25], or emergent properties such as community structure [18]. Note that a graph model is not expected to reproduce any specific graph, or to generate isomorphic graphs. Similarly, the automatic construction of graph models is not an isomorphism problem, thus producing isomorphic graphs is not the intention.

To perform the analysis of network measures, this work made use of six well-known graph models exhibiting a variety of different properties. The Growing Random (GR) model [4] demonstrated simple growth. The Barabasi-Albert (BA) [4] and Aging Preferential Attachment (APA) [9] models depicted scale-free degree distributions. The Erdos-Renyi (ER) [10] and Watts-Strogatz (WS) [25] models represented low and high clustering coefficients, respectively. Both the ER and WS models produced low average geodesic path lengths, which the latter was explicitly designed to achieve. Finally, the Forest Fire (FF) model [17] produced heavily-tailed degree distributions and community structure. Note that while all six aforementioned models were used to analyze the network measures, only the BA and ER models were used as targets for the GP system. A further discussion of graph models can be found in [15].

[1] Transitivity is also commonly referred to as the clustering coefficient.

3 Meta-Analysis of Network Properties

To determine the performance of a set of centrality measures, used to differentiate networks generated by different graph models, a method of combining single-measure results was devised as follows. For a given target graph G, graph model M, and set of measures F, N instances of M were generated. For each of the instances of M, each centrality measure was compared using a Kolmogorov-Smirnov (KS) test [19], at a 95 % confidence level, to that of the target graph, G. The power set (excluding the empty set) of F was generated to examine all non-empty subsets of measures. For each subset of measures, the p-values corresponding to its members were combined using Fisher's method [12].

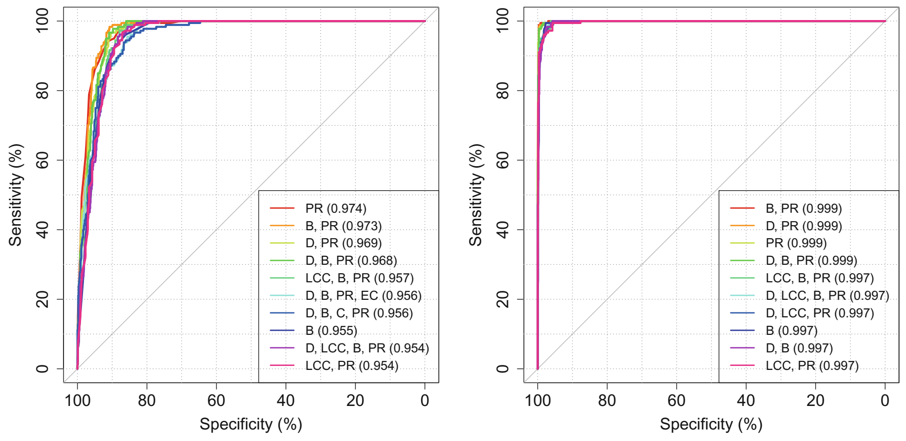
The procedure outlined above only compared a single target to a single model. To compare multiple models to a single target graph, a meta analysis procedure was used. A classification scheme was derived, allowing ROC curves to be used for analysis. To construct such a classifier, an assumption was made that if two graphs were generated by the same model, they would exhibit similar centralities and thus a high p-value would be obtained when compared. This assumption is reasonable in that a measure for which this doesn't hold must produce significantly different values for networks generated by the same model, and thus is not a good measure for determining network/model similarity. By extension of the above assumption, a good subset of measures should produce a high p-value when combined. If two graphs were produced by the same model, the combined p-value from each subset of measures was expected to be 1. Conversely, if the graphs were produced by different models, the combined p-value was expected to be 0. This reasoning was used to derive a classification system where the observed p-values from this procedure were taken as an approximation (i.e., the response) to the expected outcome.

The area under the curve (AUC) was calculated for each ROC curve as a measure of performance for the corresponding classifier. In the context of this work, the AUC represented the probability that G, originating from model M, will receive a higher p-value using Fisher's method when compared to graphs generated by M than when compared to graphs not generated by M.

3.1 Meta-Analysis Results

This section presents the results obtained by repeating the meta-analysis procedure above using a target graph generated by each of the six models and aggregating the results. One might argue that using only a single target network may not be truly representative of the model family. However, in a real-world scenario, there is often only a single example of the network being modeled. Thus, using a single target network drew a closer parallel to a real-world scenario.

Figure 1 presents the ROC curves for 100 (smallest) and 1000 (largest) vertex networks. The higher AUC values observed for 1000 vertex networks demonstrated that larger networks were easier to distinguish due to their more pronounced and emergent structural differences. A key observation was that the highest AUC attained at each network size was produced by either the set containing the PageRank measure alone or both the PageRank and betweenness



(a) 100 vertex networks (b) 1000 vertex networks

Fig. 1. ROC curves depicting the ten measure sets with the highest area under the curve (AUC) values, shown in the legend, for various network sizes.

measures. At each of the four network sizes, the PageRank measure was included in each of the top five sets of measures. Furthermore, PageRank was present in nine of the top ten measure sets for the 100, 250, and 500 vertex networks and eight of the top ten sets when 1000 vertex networks were examined. Similarly, betweenness was present in at least five of the top ten fitness sets for each network size.

When two measures were combined, it was noted that combining the betweenness and PageRank measures would insignificantly change the AUC value relative to only using the PageRank measure, namely the AUC was different by at most 0.002. When three measures were considered, the subset which contained the degree, betweenness, and PageRank measures obtained the highest AUC. As the number of measures combined was increased beyond three, the subsets which attained the highest AUC became less intuitive. However, it was noted that the subsets which obtained the highest AUC always contained the degree, betweenness, and PageRank measures. Based on these observations, the degree, betweenness, and PageRank were chosen as measures of evolutionary fitness to be used by the GP system, detailed in the following section.

4 Automatic Construction of Graph Models

To demonstrate the effectiveness of the identified centrality measures as evolutionary fitness criteria, the LinkableGP system [20,21], a linear-object-oriented GP, was used to automatically construct graph models. An abstract class representing a generalized graph model, based on the model given in [20], was provided to the system to define the structure of the evolved models. This abstract model consisted of three unimplemented methods, namely *SelectVertices*, *CreateEdges*, and *SecondaryActions*, which were evolved by the GP system. The generalized model, beginning with an initially empty graph, constructed a network using a single loop, executed once for each vertex to be created. First, *SelectVertices* was executed, returning a collection of vertices, C, as potential candidates for the new vertex to attach to. For each vertex in C, *CreateEdges* was executed to produce a list of edges followed by *SecondaryActions*, which could add new vertices to C. Each of the evolved methods were provided their own language elements as detailed in the subsequent sections.

4.1 SelectVertices Method

For each of the methods listed below, a collection which prevented previously seen elements from being inserted was returned in either stack or queue form.

- **GetAll{Stack, Queue}(g)** – Select all vertices from g.
- **GetRandom{Stack, Queue}(g)** – Select a random vertex from g.
- **GetRandom{Stack, Queue}(g, n)** – Select n random vertices from g.
- **GetRoulette{Stack, Queue}(g, f)** – Select a vertex from g using roulette selection with probabilities assigned to vertices via the evaluator function f.
- **GetRoulette{Stack, Queue}(g, f, n)** – Select n vertices from g using roulette selection as above.

The vertex evaluator functions and composition operators below were made available to facilitate more robust selection procedures. In addition to these functions, integer and floating-point generators with (inclusive) ranges $(1, 10)$ and $(0.0, 1.0)$, respectively.

- **GetDegree( )** – Computes the degree of the vertex.
- **GetLocalTransitivity( )** – Computes the local transitivity of the vertex.
- **GetAge( )** – Computes the age of the vertex.
- **Add(f1, f2)** – Computes $f_1(v) + f_2(v)$.
- **Add(f, d)** – Computes $f(v) + d$.
- **Mult(f1, f2)** – Computes $f_1(v) \times f_2(v)$.
- **Mult(f, d)** – Computes $f(v) \times d$.
- **Pow(f, d)** – Computes $f(v)^d$.
- **InversePow(f, d)** – Computes $f(v)^{-d}$.

4.2 CreateEdges Method

Each of the functions made available to the *CreateEdges* method, shown below, returned a list of edges which were added to the graph after the secondary selection took place to prevent interference with the *SecondaryActions* procedure. A floating-point generator with an (inclusive) range $(0.0, 1.0)$ was also provided.

- **AddEdge(v1, v2)** – Return an edge between *v1* and *v2*.
- **EmptyEdge( )** – Designates no edge was to be created.
- **AddEdgeWithProb(v1, v2, p)** – With probability p, return an edge between *v1* and *v2*.
- **AddTriangle(v1, v2)** – Return two edges forming a triangle including *v1*, *v2* and a randomly chosen neighbor of *v1* or *v2*.

- **AddTriangleWithProb(v1, v2, p)** – Create an edge between *v1* and *v2*, and with probability *p*, create a triangle (as above).
- **Duplicate(v1, v2, p)** – Returns a list of edges between *v1* and each neighbor of *v2*. With probability *p*, an edge is also created between *v1* and *v2*.

4.3 SecondaryActions Method

The *SecondaryActions* method was responsible for performing actions as a direct result of adding a vertex and/or edge(s). Integer and floating-point constant generators with (inclusive) ranges $(0, 10)$ and $(0.0, 1.0)$, respectively, were also available.

- **AddNeighbours(c, n, v)** – Adds *n* randomly selected neighbors of vertex *v* to the taboo collection *c*.
- **GetRandomValue(a, b)** – Returns a uniformly random integer between *a* and *b*, inclusive, where *a* and *b* are integer arguments.
- **GetGeometricValue(p)** – Returns an integer generated according to a geometric distribution with probability *p*.

5 Results and Discussion

By evolving a graph model for a known algorithm, the evolved model can be easily validated against both the network used as a target, referred to as the *target graph*, as well as other networks generated by the model. Two well-known models, namely the BA and ER models were selected as target models. The BA model was selected as it produces scale-free, power-law degree distributions which are commonly found in real-world networks. While the BA model, as originally proposed, is limited in the degree distributions it can generate, the model nonetheless describes many real world networks such as actor affiliation networks and the World Wide Web [4]. The ER model was selected as it can exhibit non-zero clustering coefficients and does not produce a static number of edges each iteration. Furthermore, these two models have been used in previous works [2,3,22]. For this study, the BA model created a single edge per iteration and used linear preferential attachment while the connection probability of the ER model was set to 0.05 (5 %) to prevent excessive edge density.

For both models, a target graph was generated with 100 and 500 vertices. For each target graph, the GP system was run 30 times to produce a set of candidate models using empirically-determined parameters as follows. A population of 50 individuals was evolved over 50 generations and used an 80 % crossover rate and 20 % mutation rate. Tournament selection using 3 individuals was employed and elitism was set at 2 individuals per generation. Initial chromosome lengths were randomly assigned within the following ranges: 1–15 for the *SelectVertices* and *SecondaryActions* methods whereas the *CreateEdges* method had a smaller range of 1–5. A sum-of-ranks strategy was employed during evolution using three fitness measures, namely the KS test statistic comparing the degree distribution, betweenness centrality, and PageRank measures of a single network generated by the candidate model and the target network. The final model was selected as the highest ranked model from the 30 runs using sum-of-ranks. In addition to the centrality measures used during evolution, the number of edges, average geodesic path length (AGP), and global clustering coefficient (CC) were also compared between networks generated by the evolved model and the target model. Bold KS statistic entries denote the average test statistic was below the critical threshold.

5.1 Evolving the Barabasi-Albert Model

When a 100 vertex BA network was used as the target, both the mean and minimum observed AGP, shown in Table 1, were higher for the evolved model than the true model. Although the mean and minimum AGPs were 0.325 and 0.352 higher, respectively, both were within a single standard deviation of the true model. By definition, the transitivity of both the evolved and BA models was zero and the number of edges was constant. Examining the centrality measures, the average D statistic for the PageRank measure (0.159) was relatively high compared to the degree and betweenness measures (at 0.053 and 0.066 respectively), however, this value was still below the critical threshold of 0.192.

When a 500 vertex BA network was used as the target, the average AGP was 0.261 higher in graphs generated by the evolved model than the true model while the maximal difference was 0.776. However, examining the centrality measures demonstrated that the evolved and true models produced similar networks with respect to the employed fitness measures. With average KS statistics of 0.023, 0.026, and 0.073 for the degree, betweenness, and PageRank measures, respectively, the centrality measures were insignificantly different among networks generated by the evolved and true models. For comparison, Algorithm 2 presents the evolved BA_{500} model, simplified to remove bloat, alongside the true BA model (Algorithm 1). Note that in the true model, the in-degree was used in a directed fashion which caused slightly different selection probabilities.

In summary, the BA model was effectively reproduced in both experiments, however, the evolved models demonstrated higher values for the AGP measure.

Algorithm 1. *BA* Model

```
function SelectVertices($g$)
    $f \leftarrow$ GetInDegree( )    //Degree - 1
    $S \leftarrow$ GetRouletteStack($g, f$)
    return $S$
end function
function CreateEdges($v, u$)
    $E \leftarrow$ AddEdge($u, v$)
    return $E$
end function
function SecondaryActions($v, S$)
        //No action
end function
```

Algorithm 2. Simplified BA_{500} Model

```
function SelectVertices($g$)
    $f \leftarrow$ GetDegree( )
    $S \leftarrow$ GetRouletteStack($g, f$)
    return $S$
end function
function CreateEdges($v, u$)
    $E \leftarrow$ AddEdge($u, v$)
    return $E$
end function
function SecondaryActions($v, S$)
        //No action
end function
```

Table 1. Comparison of networks generated by the evolved models and their respective true models.

Vertices		Measure	Min	μ	Max	σ
100	BA_{100}	Edges	99	99	99	0
		AGP	3.947	4.806	5.843	0.481
		CC	0	0	0	0
	Barabasi-Albert	Edges	99	99	99	0
		AGP	3.595	4.481	5.842	0.522
		CC	0	0	0	0
	Average D Statistic	Degree	**0.020**	**0.053**	**0.100**	0.020
		Betweenness	**0.030**	**0.066**	**0.140**	0.024
		PageRank	**0.090**	**0.159**	0.300	0.043
500	BA_{500}	Edges	499	499	499	0
		AGP	5.497	6.353	7.640	0.560
		CC	0	0	0	0
	Barabasi-Albert	Edges	499	499	499	0
		AGP	5.261	6.092	6.864	0.434
		CC	0	0	0	0
	Average D Statistic	Degree	**0.010**	**0.023**	**0.050**	0.011
		Betweenness	**0.012**	**0.026**	**0.050**	0.009
		PageRank	**0.050**	**0.073**	0.114	0.016
100	ER_{100}	Edges	214	246.833	280	13.643
		AGP	2.855	3.013	3.237	0.084
		CC	0.026	0.048	0.070	0.012
	Erdos-Renyi	Edges	217	245.867	267	14.920
		AGP	2.900	3.033	3.221	0.102
		CC	0.024	0.051	0.067	0.010
	Average D Statistic	Degree	**0.030**	**0.099**	0.220	0.041
		Betweenness	**0.060**	**0.107**	**0.160**	0.030
		PageRank	**0.060**	**0.106**	**0.140**	0.024
500	ER_{500}	Edges	6529	6660.767	6761	58.078
		AGP	2.164	2.175	2.189	0.006
		CC	0.051	0.053	0.054	0.001
	Erdos-Renyi	Edges	6094	6238.300	6366	62.745
		AGP	2.209	2.223	2.240	0.007
		CC	0.048	0.050	0.052	0.001
	Average D Statistic	Degree	**0.080**	0.145	0.224	0.033
		Betweenness	**0.044**	**0.069**	0.090	0.013
		PageRank	**0.036**	**0.056**	0.088	0.011

5.2 Evolving the Erdos-Renyi Model

When a 100 vertex ER network was used as a target, the evolved model showed a connection probability of 0.0507 – a difference of 0.0007 compared to the true model. The post-validation results, shown in Table 1, showed that the average number of edges, AGP, and CC measures were similar between the evolved model and the true model; the evolved model exemplified 0.0957 more edges, 0.020 lower AGP, and 0.003 lower CC, on average, than the true model. The average KS statistics at 0.099, 0.107, and 0.106 for the degree, betweenness, and PageRank measures, respectively, were all well below the critical threshold of 0.192 which further demonstrated the similarity among the evolved and true models.

Algorithm 3. *ER* Model

```
function SelectVertices($g$)
    $S \leftarrow$ GetAllStack($g$)
    return $S$
end function
function CreateEdges($v, u$)
    $E \leftarrow$ AddEdgeWithProb($v, u, 0.0500$)
    return $E$
end function
function SecondaryActions($v, S$)
        //No action
end function
```

Algorithm 4. Simplified ER_{500} Model

```
function SelectVertices($g$)
    $S \leftarrow$ GetAllStack($g$)
    return $S$
end function
function CreateEdges($v, u$)
    $E \leftarrow$ AddEdgeWithProb($v, u, 0.0535$)
    return $E$
end function
function SecondaryActions($v, S$)
        //No action
end function
```

When a 500 vertex ER network was used as the target, the evolved model had a 0.0535 connection probability and exemplified roughly 422 more edges per graph, on average. As Table 1 demonstrated, networks produced by the evolved model had an AGP that was 0.048 higher than those produced by the true model, while the average transitivity was only 0.003 higher. The significantly different degree distributions (average KS statistic of 0.145) among networks generated by the true and evolved models were attributed to the increased connection probability, and therefore the increased expected degree, of the evolved model. Conversely, the average KS statistic for the betweenness and PageRank measures, 0.069 and 0.056, respectively, were both well below the critical threshold of 0.086. For comparison, Algorithm 4 presents the evolved ER_{500} model, simplified to remove bloat, alongside the true ER model (Algorithm 3).

In summary, the GP system was able to reproduce the ER model with striking accuracy, however, slightly higher connection probabilities were evolved.

6 Conclusion

This paper proposed a meta-analysis framework to analyze the discriminatory power of centrality measures when comparing graph models of complex networks.

Six well-known graph models were used to evaluate six network centrality measures. Results indicated that of the examined centrality measures, the degree distribution, betweenness centrality, and PageRank were the most effective for quantifying the (dis)similarity of networks generated by different graph models. A genetic programming (GP) system for the automatic construction of graph models was proposed using the results of the meta-analysis to form the fitness evaluation. The GP system was used to automatically infer two well-known graph models, namely the Barabasi-Albert (BA) and Erdos-Renyi (ER) models. Target networks from these models were generated with 100 and 500 vertices, respectively, and used as the target networks within the GP system. Results indicated that these well-known graph models could be evolved with striking accuracy. Furthermore, the exceptional quality of the evolved models provided empirical evidence of the proposed meta-analysis' merit.

Many avenues of future study have become apparent throughout the course of this work. First and foremost, this paper only addresses undirected, unweighted networks and considers only centrality measures. Examining further, non-centrality network measures along with weighted and directed networks should be an immediate future study.

References

1. Arianos, S., Bompard, E., Carbone, A., Xue, F.: Power grid vulnerability: a complex network approach. Chaos: an Interdisciplinary. J. Nonlinear Sci. **19**(1), 013119 (2009)
2. Bailey, A., Ventresca, M., Ombuki-Berman, B.: Genetic programming for the automatic inference of graph models for complex networks. IEEE Trans. Evol. Comput. **18**(3), 405–419 (2014)
3. Bailey, A., Ventresca, M., Ombuki-Berman, B.: Automatic generation of graph models for complex networks by genetic programming. In: Proceedings of the Fourteenth International Conference on Genetic and Evolutionary Computation Conference, GECCO 2012, pp. 711–718. ACM, New York, NY, USA (2012)
4. Barabási, A.L., Albert, R.: Emergence of scaling in random networks. Science **286**(5439), 509–512 (1999)
5. Berg, J., Lässig, M., Wagner, A.: Structure and evolution of protein interaction networks: a statistical model for link dynamics and gene duplications. BMC Evol. Biol. **4**(1), 51 (2004)
6. Bonacich, P.: Power and centrality: a family of measures. Am. J. Sociol. **92**, 1170–1182 (1987)
7. Brin, S., Page, L.: The anatomy of a large-scale hypertextual web search engine. Comput. Netw. ISDN Syst. **30**(1), 107–117 (1998)
8. Chung, F., Lu, L.: The average distance in a random graph with given expected degrees. Internet Math. **1**(1), 91–113 (2004)
9. Dorogovtsev, S.N., Mendes, J.F.F.: Evolution of networks with aging of sites. Phys. Rev. E **62**(2), 1842 (2000)
10. Erdös, P., Rényi, A.: On random graphs. Publicationes Mathematicae **6**, 290–297 (1959)
11. Fan, Z., Chen, G., Zhang, Y.: Using topological characteristics to evaluate complex network models can be misleading. arXiv preprint arXiv:1011.0126 (2010)

12. Fisher, R.: Statistical Methods for Research Workers. Oliver and Boyd, Edinburgh (1925)
13. Freeman, L.C.: A set of measures of centrality based on betweenness. Sociometry **40**, 35–41 (1977)
14. Freeman, L.C.: Centrality in social networks: conceptual clarification. Soc. Netw. **1**(3), 215–239 (1979)
15. Goldenberg, A., Zheng, A.X., Fienberg, S.E., Airoldi, E.M.: A survey of statistical network models. Found. Trends Mach. Learn. **2**(2), 129–233 (2010)
16. Leskovec, J., Chakrabarti, D., Kleinberg, J., Faloutsos, C., Ghahramani, Z.: Kronecker graphs: an approach to modeling networks. J. Mach. Learn. Res. **11**, 985–1042 (2010)
17. Leskovec, J., Faloutsos, C.: Scalable modeling of real graphs using Kronecker multiplication. In: Proceedings of the 24th International Conference on Machine Learning, pp. 497–504. ACM (2007)
18. Leskovec, J., Kleinberg, J., Faloutsos, C.: Graphs over time: densification laws, shrinking diameters and possible explanations. In: Proceedings of the eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining, pp. 177–187. ACM (2005)
19. Massey Jr., F.J.: The kolmogorov-smirnov test for goodness of fit. J. Am. Stat. Assoc. **46**(253), 68–78 (1951)
20. Medland, M.R., Harrison, K.R., Ombuki-Berman, B.: Demonstrating the power of object-oriented genetic programming via the inference of graph models for complex networks. In: 2014 Sixth World Congress on Nature and Biologically Inspired Computing (NaBIC), NaBIC 2014, pp. 305–311. IEEE (2014)
21. Medland, M.R., Harrison, K.R., Ombuki-Berman, B.: Incorporating expert knowledge in object-oriented genetic programming. In: Proceedings of the 2014 Conference on Genetic and Evolutionary Computation Companion, GECCO Comp 2014, pp. 145–146. ACM, New York, NY, USA (2014)
22. Menezes, T., Roth, C.: Symbolic regression of generative network models. Scientific reports 4 (2014)
23. Newman, M.: Networks: An Introduction. Oxford University Press, Oxford (2010)
24. Stam, C.J., Reijneveld, J.C.: Graph theoretical analysis of complex networks in the brain. Nonlinear Biomed. Phys. **1**(1), 3 (2007)
25. Watts, D.J., Strogatz, S.H.: Collective dynamics of small-world networks. Nature **393**(6684), 440–442 (1998)