Noise-Tolerant Model Selection and Parameter Estimation for Complex Networks

Sadegh Aliakbary[a,*], Sadegh Motallebi[a], Sina Rashidian[a], Jafar Habibi[a], Ali Movaghar[a]

[a]*Department of Computer Engineering, Sharif University of Technology, Tehran, Iran*

Abstract

Real networks often exhibit nontrivial topological features that do not occur in random graphs. The need for synthesizing realistic networks has resulted in development of various network models. In this paper, we address the problem of selecting and calibrating the model that best fits a given target network. The existing model fitting approaches mostly suffer from sensitivity to network perturbations, lack of the parameter estimation component, dependency on the size of the networks, and low accuracy. To overcome these limitations, we considered a broad range of network features and employed machine learning techniques such as genetic algorithms, distance metric learning, nearest neighbor classification, and artificial neural networks. Our proposed method, which is named ModelFit, outperforms the state-of-the-art baselines with respect to accuracy and noise tolerance in different network datasets.

Keywords: Complex Networks, Network Models, Model Selection, Parameter Estimation, Machine Learning, Social Networks

1. Introduction

Many systems in the real world can be modeled as networks. Real-world networks often exhibit non-trivial topological features which are missing in random graphs [1]. Such graphs, which are called complex networks, appear in social networks, biological networks, and many other domains. In the past two decades, network models are proposed in the literature to synthesize realistic networks [2, 3, 4, 5, 6]. The artificial networks are supposed to follow the non-trivial topological features of real networks. For example, Barabási-Albert model [2] synthesizes scale-free networks with long-tail degree distribution, and Watts-Strogatz model [3] generates small-world networks with high clustering. Despite the advances in the field, there is no universal network model suitable for all applications. Thus, a prerequisite of network generation is the step of model selection. Additionally, appropriate parameters for the selected model should be estimated. However, model fitting is not a trivial task, and intelligent methods are required for model selection and parameter estimation.

In any application that involves network generation, we should select an appropriate network model and then tune its parameters in order to generate networks with desired properties. Particularly, it is often required to fit a model to a target network instance, or to fit it to a set of target feature values. In this case, the fitted model is supposed to synthesize networks similar to the target network. The "model fitting" process aims at finding the best model being capable of generating networks similar to the target network (*Model Selection*), and then estimating the model parameters which generate the most similar graphs to the target network (*Parameter Estimation*). Applications of network model fitting include simulation of network dynamics [7, 8, 9], graph summarization [10, 5, 11, 12, 13], network sampling [14, 15, 16, 17, 18], network anonymization [19, 20, 21, 22], and network comparison [23, 10, 24, 25, 26, 27, 11].

[*]Corresponding author

Email addresses: aliakbary@ce.sharif.edu (Sadegh Aliakbary), motallebi@ce.sharif.edu. (Sadegh Motallebi), rashidian@ce.sharif.edu. (Sina Rashidian), jhabibi@sharif.edu (Jafar Habibi), movaghar@sharif.edu (Ali Movaghar)

Most of the existing model fitting methods lack the parameter estimation component and only provide a model selection method (e.g., [28, 29, 30, 31]). Additionally, the existing methods are sensitive to network perturbations, and their accuracy drops considerably if we inject noise to the network by making random changes in network edges. However, noise tolerance is an important requirement for model fitting methods. This is because, in most cases, the target network (e.g., a real network) is fully compatible with no network model and shows a level of difference (noise) to any candidate model.

In this paper, we propose an intelligent and noise-tolerant model fitting method for complex networks, which is named *ModelFit*. ModelFit consists of both model selection and parameter estimation components. It utilizes a diverse set of local and global network features, and employs various machine learning algorithms for model selection and parameter estimation. The model selection component of ModelFit is based on genetic algorithms, distance metric learning, and nearest neighbor classification. The parameter estimation is performed by learning an artificial neural network. As a result, ModelFit is become an accurate, robust-to-noise, and size-independent method.

The structure of the paper in the following sections is as follows: Section 2 reviews the literature on the problem of model fitting. In section 3 we propose an intelligent method for model selection. The comprehensive evaluations of the proposed model selection method is presented in section 4 along with comparison to baseline methods. In section 5, a parameter estimation method is proposed and the accuracy of the method is evaluated. Finally, we conclude the paper in section 8.

2. Related Works

The existing model fitting methods follow different approaches and techniques. The set of considered network features is a source of diversity in the existing methods. Although some methods use both local and global network features in model selection [28, 13], many existing methods are based on graphlet-counting features [29, 30, 31, 25, 26]. Graphlet counting is an inefficient approach, often accelerated by sampling [29] or approximation algorithms [32, 29] which result in accuracy drops [28].

A network model can be regarded as a class of networks, and model selection is actually a classification problem in which the target network is assigned to one of the candidate models. The classification method is another source of difference among the existing model selectors. Some existing model selection methods are based on supervised machine learning algorithms of classification, such as decision tree learning [28, 29, 30]. Some other methods are based on lazy learning and distance-based classification [31]. In this approach, the target network is compared with a set of generated networks, and the most similar model is selected. In this process, the model of the target network is selected either based on its average distance to networks of different models (e.g., [31]) or according to the class of the nearest neighbors (e.g., our proposed method). It is also worth noting that some of the existing model fitting methods are sensitive to the size of the target network (e.g., [29]), or dependent on network density (e.g., [28, 29, 30]).

3. Model Selection

In this section, we describe our proposed method for model selection. Given a target network instance, or its structural features, ModelFit finds the best model that is able to generate similar networks. ModelFit selects the best fitting class (model) based on the distance of the target network to various generated network instances. In the proposed method, a network distance metric is first learned which is able to separate networks of different classes. In this way, a genetic algorithm (GA) is utilized as an intelligent search method in order to find the best network distance metric. The resulting distance function, called NetDistance, is then utilized in nearest neighbor classification of the networks. In this phase, different network instances are generated using various network models, and among them, the most similar instances to the target network are selected. This process describes the well-known “k nearest neighbor (KNN)” classification algorithm. KNN is a simple machine learning algorithm that classifies an object by a majority vote of its *k* neighbors. Figure 1 shows the methodology of ModelFit for model selection.

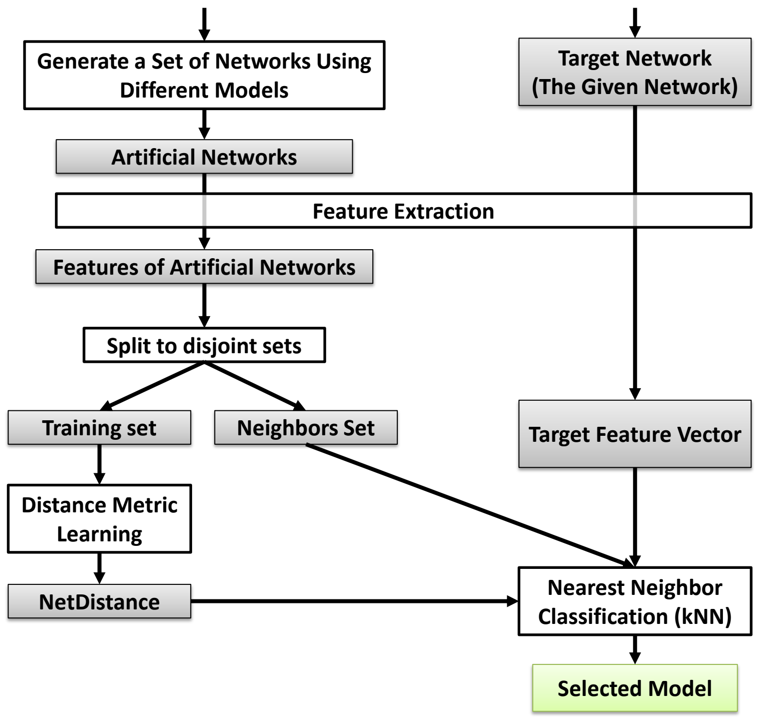


Figure 1: The proposed methodology for model selection

3.1. Network Features

In the proposed method, the structural properties of each network instance is translated into a set of numerical features. Various measures are defined in the literature to quantify the structural properties of networks. We utilized a diverse set of local and global network features in forming the network feature vectors. The considered network features are: Average shortest path length [33], effective diameter [6], density [34], average degree [33], clustering coefficient [3], transitivity [33], modularity [35], assortativity (degree correlation) [36], and the degree distribution [37, 38]. Although many other network features are also illustrated in the literature, the integration of the selected features resulted in an accurate and robust to noise model selector, and the proposed method experienced no more improvement by considering other features. In this research, we have only considered simple graphs, and measurements related to directed and weighted graphs are not investigated.

3.2. Distance Metric Learning and Model Selection

The proposed model selection method (ModelFit) is based on a network distance metric (NetDistance) which is able to separate networks with different models. In this subsection, we describe the machine learning process which resulted in NetDistance function.

In order to learn the NetDistance function, we first generate a set of networks (*Training Set*) using different network models. The networks in the training set are labeled by the corresponding models. Each network is represented by a feature vector, and the distance between two networks is supposed to be computed as a function of the two feature vectors. We utilized a genetic algorithm [39] to learn the NetDistance function for network comparison. We considered the weighted Manhattan distance (Equation 1) as the template of the distance function, since it outperformed alternatives such as weighted Euclidean distance (Equation 2) and weighted Canberra distance (Equation 3) in our experiments. As a result, the distance metric learning problem is transformed to a “weight assignment” to the features in the weighted Manhattan distance. In Equations 1-3, p and q show the feature vectors of the compared networks,

p_i and q_i represent the i^{th} extracted feature from network p and q respectively, and w_i is the assigned weight to the i^{th} feature. Finding an appropriate set of feature weights in the distance function is a search problem with a very large search space, and genetic algorithms are capable of solving such search problems efficiently [39]. A genetic algorithm is configured based on the chromosome representation and the definition of the fitness function. We represent a chromosome by a vector of some real-valued weights corresponding to network features. In the natural selection process of the genetic algorithm, the feature weights evolve which results in best obtainable weights. We also define the fitness function based on the ability of the distance metric to classify the network instances. We employed the distance function in k-nearest-neighbour (KNN) algorithm and the accuracy of the KNN classifier is utilized as the fitness function. The accuracy of the distance function d is described in Equation 4, in which $knnclassify_d(n)$ shows the predicted class of network n by a KNN classifier that utilizes d as the distance function, and $class(n)$ is its actual class. As a result, those chromosomes that best classify the networks of the training data survive in the population. The described intelligent search method results in a weighted Manhattan distance function, which is called *NetDistance*.

$$d_M(p,q) = \sum_{i=1}^{n} |w_i p_i - w_i q_i| \tag{1}$$

$$d_E(p,q) = \sqrt{\sum_{i=1}^{n} (w_i q_i - w_i p_i)^2} \tag{2}$$

$$d_C(p,q) = \sum_{i=1}^{n} w_i \frac{|p_i - q_i|}{|p_i| + |q_i|} \tag{3}$$

$$acc(d) = P(knnclassify_d(n) = class(n));\ n \in network\ dataset \tag{4}$$

Having NetDistance as the network distance function, we use it in a nearest neighbor classification algorithm in order to find the model that best fits the target network. In this way, we first generate many network instances (the neighbors set) using different network models, and then we compare the generated networks with the target network to find the most similar model. As Figure 1 shows, we utilize KNN which is an instance-based (lazy) learning algorithm for classification. The KNN classifier uses the NetDistance function to compare the target network with the generated networks (neighbors set). The neighbors set and the training set are disjoint, to ensure a fair evaluation.

4. Evaluation of the Model Selector

In this section, we evaluate the proposed model selection method. The evaluations include the creation of network datasets, comparison of the proposed approach with notable baseline methods, and examination of the noise effect on different methods.

4.1. Dataset Description

In order to evaluate the accuracy of a model selection method, we use it for predicting the model of some labeled network instances. Model selection is actually a classification problem. Therefore, we can evaluate the model selector based on a dataset of networks with known models. We consider six network models in our evaluations: Barabási-Albert model [2], Erdős-Rényi [40], Forest Fire [6], Kronecker model [5], random power-law [4], and Small-world (Watts-Strogatz) model [3]. To the best of our knowledge, these models are amongst the most widely used methods in network generation applications, and they also cover a wide range of network structures.

A network model offers a set of parameters for tuning the synthesized networks so that they follow the target properties. The network size (number of nodes) is the common parameter of the considered models. In our evaluations, the network size parameter is randomly selected from 1,000 to 5,000 nodes. For any network model, other parameters are also selected randomly from the parameter ranges, as described in Appendix A.

We designed the evaluations in 10 rounds. In each round, 600 different artificial networks are synthesized by the means of the six described models (100 networks per model), using randomly chosen parameters. 90 percent of these generated networks (540 instances) are randomly selected as the *training set* which are used in learning NetDistance function, and the remaining 60 networks are used either in the *neighbors set* or as the *target network* (see Figure 1). Each evaluation round includes 60 iterations. In each iteration, one instance out of the 60 networks is used as the *target network*, and the remaining networks are used in the *neighbors set*. The KNN classifier compares the target network with the neighbors set, using the trained NetDistance function, and selects a model for the target network based on the votes (models) of its most similar networks in the neighbors set. If the predicted model is equal to the actual model of the target network, the score of the method in the evaluation is increased. The accuracy of the method is finally calculated as its average score in different rounds and iterations.

In each iteration we also include five sub-iterations in order to consider various noise levels (noise=5, 10, 15, 20, and 25 percent). The noise in a network is simulated by randomly rewiring a fraction of its edges. For example, the 5% noise level is achieved by rewiring five percent of the network edges to randomly chosen pair of nodes. In each evaluation iteration, the noise is injected into the neighbors set and the target network. As a result, in each of the 10 evaluation rounds, 900 networks are generated: 600 pure network instances plus 300 noisy networks (60 noisy instances per noise levels). Thus, the evaluations resulted in a dataset with a total of 9000 different artificial networks, generated by randomly chosen parameters.

We have also utilized the network dataset presented in the GMSCN method [28] in one of our experiments (refer to subsection 4.4) in order to show that the reported results are not biased toward our prepared dataset. However, we should note that the prepared dataset has some advantages over the GMSCN's dataset. For example, GMSCN is restricted to networks with similar densities. Additionally, our dataset is much larger than the GMSCN dataset and thus provides more reliable results.

4.2. Baseline Methods

In order to evaluate our proposed ModelFit method, we compare it with different alternative model selection approaches as the baseline methods. In this subsection, we introduce the chosen baseline methods and discuss the reasons behind their selection. As discussed earlier, different approaches exist in the literature for model selection. Supervised machine learning methods of classification [41] and average-distance based classification [25, 26, 31] are some of the common approaches of model selection. We will consider representatives for all approaches in the evaluations.

Supervised machine learning algorithms may be used for network classification in order to select the best fitting model. We examined Support Vector Machines (SVM) [42, 41], the C4.5 decision tree learning [41], and artificial neural network learning [41] as the classification methods, where SVM outperformed the other alternatives. Therefore, SVM classification (called **SVMFit**) is used as a baseline method in our evaluations, and it is a representative of classification methods which are based on supervised learning. Additionally, **GMSCN** method [28] is considered as an existing model selection method which utilizes LADTree decision tree learning on a different set of network features. GMSCN is an important baseline, since it is shown to be a size-independent and accurate model selection method which outperforms some existing methods such as [29]. In the category of existing distance-based model selection methods, **RGF**-distance [25] is included in the baseline methods. RGF [25] and GDD-agreement [26] distance functions, which are based on the graphlet count features, are proposed for comparison of biological networks and are implemented in the GraphCrunch2 tool [31]. GraphCrunch2 also provides distance-based model selection methods based on RGF and GDD distance functions. We utilized the GraphCrunch2 tool for calculating the network distances. RGF outperformed GDD (both GDD amean and GDD gmean metrics [31]) in our experiments and therefore, we excluded GDD from the baselines. Additionally, nearest-neighbor classification based on RGF showed better results than those of average-distance based classification. Accordingly, the RGF baseline is implemented using a KNN (with

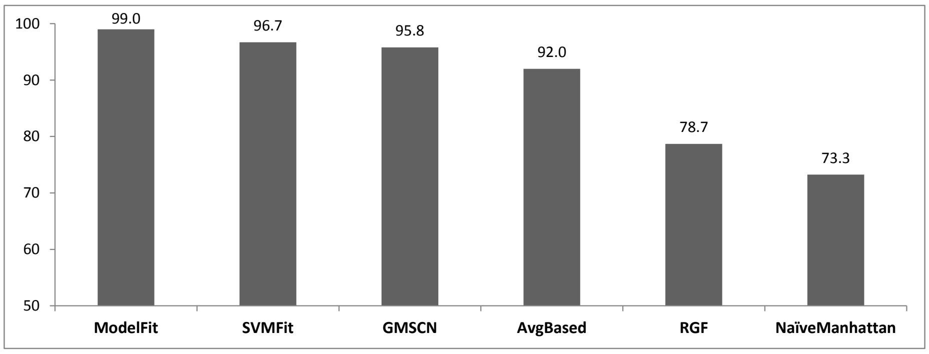


Figure 2: The accuracy of model selection methods

k=1) classifier which utilizes the RGF function as the distance function. As another baseline, **Naïve-Manhattan** distance is defined based on the pure Manhattan distance of the network features, where all the network features are equally weighted in the distance function. Euclidean and Canberra distance functions are outperformed by Manhattan distance in our experiments, and thus are excluded from the baselines. Comparing naïve Manhattan baseline with our proposed method shows that machine learning helps improve the accuracy of the distance function by assigning appropriate weights to different features. Finally, the **AvgBased** baseline method is a distance-based classifier which utilizes the learned NetDistance function, but instead of nearest-neighbor classification considers the average distances. In other words, AvgBased baseline computes the distance of the given network to all the "neighbors set" networks, then calculates its average distance to different network models, and finally selects the closest model. This baseline is included to show that despite some proposals in the literature (e.g., [31]), nearest-neighbor classification is better than average-distance classification in this application.

4.3. Implementation Details

As described in the Section 4.1, the training set consists of 540 individuals (generated networks) that evolve in the process of the genetic algorithm. An individual represents a chromosome by a set of real number values as the feature weights. In each step, a new generation of the population is produced using the crossover and mutation operations. The crossover operation selects the feature weights from one of the parent chromosomes randomly, favouring the better (more fitted) parent with probability 0.68. The mutation is performed by assigning a random value to a random feature weight, with the mutation rate of 0.15. The fitness of a chromosome is defined as the precision of the nearest neighbor classifier which uses a weighted Manhattan distance as the distance function with the weights encoded in the considered chromosome (see Equation 1). The maximum iterations for the genetic algorithm is also set to 200. In our implementation, the model selection is performed based on nearest neighbor classification. In other words, the k values is set equal to 1 in the KNN classification algorithm.

4.4. Experimental Results

The proposed method of model selection is evaluated in different scenarios, and the results of the evaluations are reported in this subsection. Figure 2 shows the accuracy of the proposed model selection method (ModelFit) along with the baseline methods. As the figure shows, ModelFit shows the best accuracy. This figure also shows the effectiveness of machine learning in this problem, since RGF and NaïveDistance, which do not utilize learning algorithms, offer the worst accuracy among the considered methods.

Figure 3 shows the results of the second experiment, in which the methods are evaluated in different noise levels. As the figure exhibits, ModelFit is the best classifier in different noise levels. Thus, ModelFit is an accurate and robust to noise model selection method. As the figure reveals, the accuracy of the distance-based methods (ModelFit, AvgBased, RGF, NaïveManhattan) drops slower than other methods (SVMFit, GMSCN). In other words, the

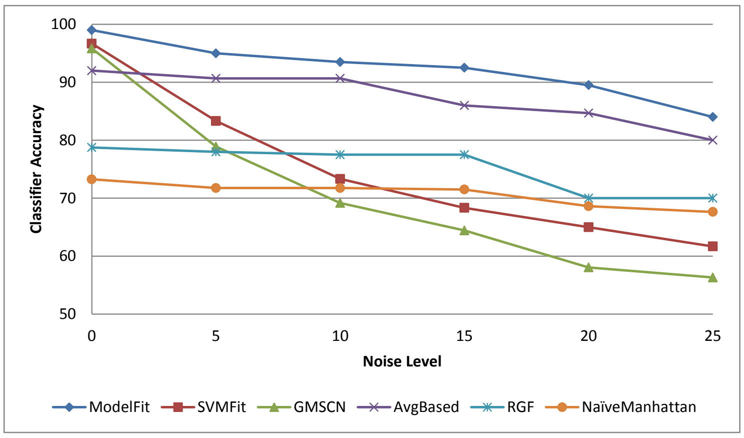


Figure 3: The noise effect on different model selection methods

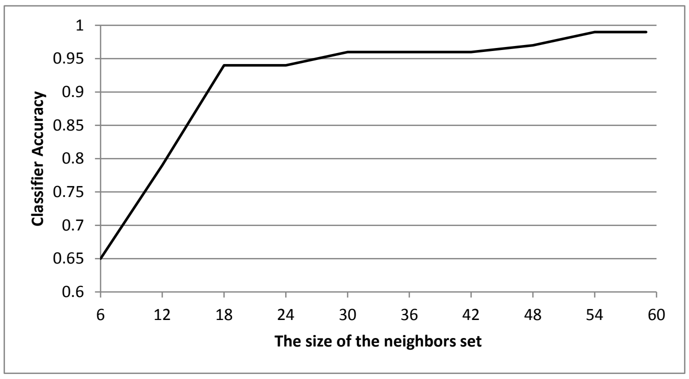
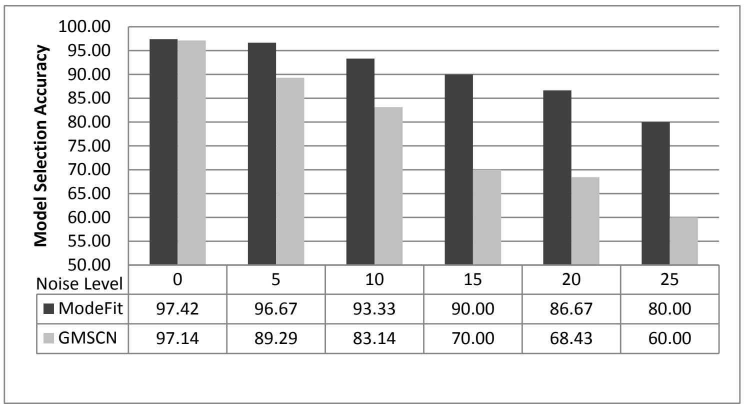


Figure 4: The effect of the neighbors set size on the accuracy of the ModelFit classifier

distance-based methods are more tolerant to the noise than those methods that utilize supervised classification learning algorithms. We should note that ModelFit also utilizes supervised machine learning algorithms in developing the NetDistance function, but ModelFit is based on learning an appropriate distance function which then helps classify the networks in a lazy learning (nearest neighbor classification) algorithm. The figure also shows that nearest neighbor classification (ModelFit) performs better than classification based on the average distances (AvgBased).

As we discussed in Section 4.1, 60 networks are utilized in each evaluation iteration (59 networks in neighbors set and one network as the target network). In the next experiment, we show that the proposed method does not necessitate a large neighbors set. Actually, with only about 50 networks in the neighbors set, an accurate model selection method is resulted. Figure 4 shows the effect of the neighbors-set size on the accuracy of ModelFit. As the figure shows, the accuracy of ModelFit increases with the size of the neighbors set and reaches a stable value with 54 networks in the neighbors set. Our experiments shows that considering more than 54 networks in the neighbors set results in no improvement in the accuracy of ModelFit.

One may argue that the results of the evaluations may be biased towards the prepared dataset. But we should note that the prepared dataset is generated independently from the proposed method, and it is a general purpose dataset which may be utilized in other research problems too. The generation parameters are also selected randomly. Moreover, the reported results are the aggregated evaluations in many different iterations on subsets of the dataset. Nevertheless, we also evaluate the proposed method based on another dataset which is presented in [28]. This dataset is generated for evaluation of the GMSCN method [28], in which 700 network instances are generated using seven generation models. The results of this evaluation is shown in Figure 5. As the figure shows, the accuracy of ModelFit in this dataset is similar to its accuracy in our prepared dataset. ModelFit also outperforms GMSCN method in this



Noise Level	0	5	10	15	20	25
■ ModeFit	97.42	96.67	93.33	90.00	86.67	80.00
□ GMSCN	97.14	89.29	83.14	70.00	68.43	60.00

Figure 5: The accuracy of the ModelFit and GMSCN model selection methods in the network dataset of GMSCN [28]

dataset too, with respect to noise tolerance. Note that this dataset consists of seven models while our prepared dataset includes six models and thus, a small accuracy drop is acceptable in the evaluations on this dataset. In GMSCN dataset, 100 network instances are generated per model (a total of 700 artificial networks). The utilized models are: Random Typing Generator (RTG) [43], Random Powerlaw [4], Kronecker graphs [5], Forest Fire [6], Preferential Attachment (Barabási-Albert) [2], Small-World Model (Watts-Strogatz) [3], and Erdös-Rényi Model [40]. In GMSCN dataset, models are configured to synthesize networks with similar densities, with an average density of networks equal to 0.0024. This is in contrast with our prepared dataset which imposes no such constraint and include a wide range of network densities.

The described evaluations confirm our intuitions for designing the model selection method: Machine learning algorithms are effective in network model selection, distance-based model selectors are robust to noise, and ModelFit does not require a large set of generated networks for predicting the target network's model. As a result, ModelFit, which utilizes machine learning algorithms to predict the model based on a learned distance function and nearest neighbor classification, outperforms alternative approaches with respect to accuracy and robustness to the noise.

5. Parameter Estimation

In this section, we investigate the parameter estimation problem and look for a model-independent approach for predicting the best generative parameters. In order to evaluate parameter estimation methods, we compare the estimated parameters with the actual values. Two criteria is measured in this comparison: The Mean Square Error (MSE) and the Pearson Correlation Coefficient (R value). MSE shows the average of the squares of the errors, where error is defined as the difference between the estimated and actual values of a parameter (see Equation 5). Pearson correlation coefficient (R value), which is defined in Equation 6, measures the linear correlation between the estimated and actual parameters. R value ranges from -1 to +1, where +1 indicates total positive correlation, 0 shows no correlation, and -1 indicates total negative correlation.

$$MSE_{XY} = \frac{1}{n}\sum_{i=1}^{n}(X_i - Y_i)^2 \tag{5}$$

$$R_{XY} = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \bar{X})^2}\sqrt{\sum_{i=1}^{n}(Y_i - \bar{Y})^2}} \tag{6}$$

As discussed earlier, ModelFit returns the closest network instance (among the neighbors set) and uses its model as the predicted model of the target network. We can also utilize the generative parameters of the closest network instance of the same model as the estimated parameters for the target network. This is an advantage of ModelFit over existing methods such as GMSCN [28] and RGF-based classification [25, 31] because those methods provide no estimation of the parameters and only predict the network model. Thus, our initial proposal is to use the generative parameters of the closest network instance of the same model as the estimated model parameters for the target network. We call this approach "Distance-based" parameter estimation. In order to evaluate the precision of the parameter estimation methods, we executed a scenario similar to the model selection evaluation which is described in Section 4. Our experiments showed that the "Distance-based" approach results in a parameter estimation with R value equal to 0.335. In other words, there is a positive correlation between the estimated and actual parameters in the "Distance-based" approach. Although this approach shows promising results, we propose another method based on learning an Artificial Neural Network (ANN) which outperforms the Distance-based approach.

In the field of machine learning, artificial neural networks (ANNs) [44] are computational models inspired by animal's brain systems which are capable of acting as universal approximators. Standard multilayer feedforward networks can approximate any measurable function to any desired degree of accuracy [44]. Therefore, ANNs are applicable in the problem of model parameter estimation. Actually, parameter estimation can be abstracted as a function that gets the network features as inputs and returns the estimated model parameters as the output. We utilize ANNs in order to learn a function that approximates the model parameters based on the network features. ANN learning requires a training set of some sample inputs and outputs of the target function. We use the generated dataset described in Section 4.1 as the training and test sets. Similar to the process of model selection evaluation, we performed 10 evaluation rounds. In each round, the 600 generated networks are utilized in a 10-fold cross-validation process, and an ANN is learned per network model. The reported results are the average outcome of the 10 evaluation rounds. The ANN is designed as a three-layer network with 10 perceptrons in the middle layer. Each perceptron in the first layer (the inputs) corresponds to a network feature value, as described in the Section 3.1. Each perceptron in the last layer (the outputs) represents the estimated value for one of the model parameters. For example, Kronecker graph model [5] includes four outputs since the 2×2 initiator matrix in this method contains four parameters to be estimated. An ANN is learned for each network model, where the topology of all of the learned ANNs are the same in the input layer and the middle layer, but the number of perceptrons in the output layer depends on the number of parameters in the network model.

When we utilized the described ANN learning method for estimating the model parameters, the pearson correlation (R value) between the estimated parameters and actual parameter values increased to 0.976, which shows a great correlation. In other words, the learned ANN can accurately estimate the generative parameters of artificial networks based on the network features. Figure 6 also shows the pearson correlation of the estimated parameters to the actual parameter values for different network models. This figure includes the R value for the distance-based approach along with the designed ANN approximation method (named ModelFit). As the figure shows, ModelFit's parameter estimations are always tightly correlated with actual values in all network models. Additionally, ModelFit outperforms the distance-based approach with respect to the R value for all models.

Table 1 shows the Mean Square Error (MSE) for ModelFit along with the distance-based method. As the table shows, ModelFit results in much smaller errors than the distance-based approach for all network models. Finally, we evaluate the accuracy of the ModelFit method for different noise levels, based on the R value criteria. Table 2 shows the correlation of the estimated parameters to actual values for different noise levels, in ModelFit method. As the table shows, the accuracy of the ModelFit estimator decreases slowly while increasing the noise level. This table also shows that even with 25% noise in the networks, ModelFit performs better than the distance-based approach: ModelFit's R value is 0.548 for 25% noise while the distance-based approach results in R value equal to 0.335 with zero noise.

We extend our experiments with a new evaluation that compares our parameter estimation method with KronFit [5], a parameter estimation method for Kronecker Graphs model. KronFit is based on maximum likelihood estimation and it maximizes the probability that the model will generate a graph identical to the original network. KronFit is a special-purpose algorithm designed for Kronfit, while ModelFit is a general-purpose parameter estimation algorithm that is applicable for any target model. Despite this fact, ModelFit outperforms KronFit for estimation of Kronecker Graph model parameters. Table 3 shows the accuracy of ModelFit and KronFit methods for estimating the actual generative parameters of the networks generated using Kronecker Graphs model [5] in our prepared dataset. As the

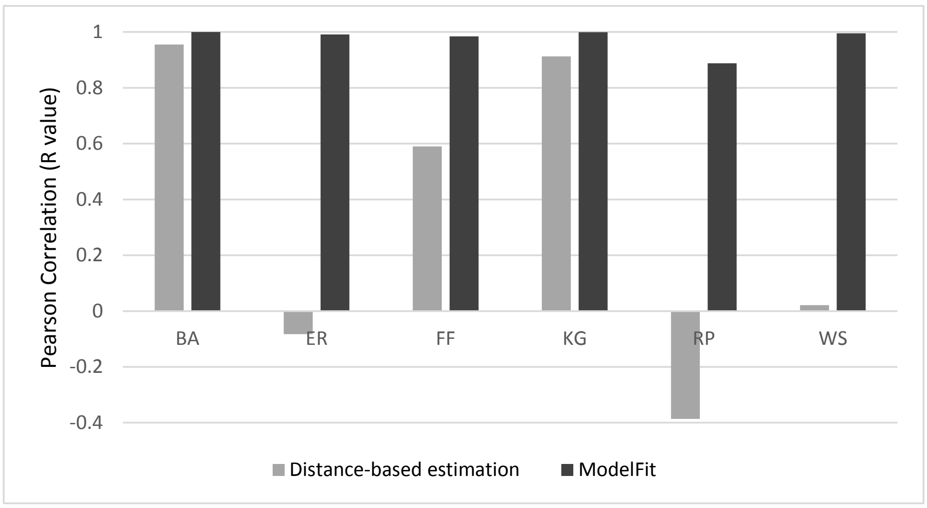


Figure 6: The Pearson correlation (R value) of the estimated parameters to the actual parameters

Table 1: The Mean Square Error (MSE) of the estimated parameters to the actual parameters

Network Model	Distance-based estimation	ModelFit
Erdős-Rényi (ER)	2.E-06	1.E-08
Barabási-Albert model (BA)	1.E+00	6.E-03
Watts-Strogatz model (WS)	5.E-06	9.E-08
Forest Fire (FF)	1.E-03	3.E-04
Kronecker graphs (KG)	4.E-04	8.E-06
Random power-law (RP)	9.E-03	6.E-03

table shows, ModelFit outperforms KronFit with respect to mean square error and Pearson correlation values.

6. Case Study

This section investigates the application of the proposed method on real networks. We first analyze the potential of NetDistance as a dissimilarity measure for comparison of real networks, and then we employ ModelFit in order to study its effectiveness for real networks.

6.1. NetDistance Dissimilarity Measure for Real Networks

NetDistance is our proposed network dissimilarity measure and the base of the proposed model fitting method. However, is NetDistance capable of explaining the structural similarities in real network data? In order to investigate

Table 2: The Pearson correlation (R value) between the actual and estimate parameters for ModelFit method in different noise levels

Noise Level	ModelFit's R value
0%	0.976
5%	0.855
10%	0.777
15%	0.713
20%	0.639
25%	0.548

Table 3: The Mean Square Error (MSE) and the Pearson correlation (R value) of the estimated parameters to the actual parameters in ModelFit and KronFit algorithms. ModelFit estimations are more accurate (lower MSE and higher R value) than those of KronFit.

	Mean Square Error (MSE)	Pearson correlation (R value)
ModelFit	8.E-06	0.99
KronFit	9.E-03	0.73

Table 4: Pairwise distances for different snapshots of *Cit_CiteSeerX* and *Collab_CiteSeerX* networks, according to NetDistance measure. NetDistance consistently shows smaller distances for networks of the same type.

	Cit_2000	Cit_2002	Cit_2004	Cit_2006	Cit_2008	Cit_2010	Collab_2000	Collab_2002	Collab_2004	Collab_2006	Collab_2008	Collab_2010
Cit_2000	0	0.048	0.099	0.115	0.132	0.139	1.565	1.475	1.363	1.366	1.381	1.326
Cit_2002	0.048	0	0.051	0.067	0.084	0.091	1.524	1.435	1.323	1.326	1.340	1.315
Cit_2004	0.099	0.051	0	0.020	0.033	0.040	1.483	1.394	1.323	1.290	1.300	1.318
Cit_2006	0.115	0.067	0.020	0	0.017	0.024	1.465	1.376	1.306	1.274	1.281	1.301
Cit_2008	0.132	0.084	0.033	0.017	0	0.007	1.457	1.362	1.306	1.268	1.267	1.301
Cit_2010	0.139	0.091	0.040	0.024	0.007	0	1.453	1.356	1.308	1.268	1.263	1.303
Collab_2000	1.565	1.524	1.483	1.465	1.457	1.453	0	0.162	0.230	0.240	0.251	0.296
Collab_2002	1.475	1.435	1.394	1.376	1.362	1.356	0.162	0	0.124	0.134	0.126	0.190
Collab_2004	1.363	1.323	1.323	1.306	1.306	1.308	0.230	0.124	0	0.053	0.084	0.066
Collab_2006	1.366	1.326	1.290	1.274	1.268	1.268	0.240	0.134	0.053	0	0.031	0.056
Collab_2008	1.381	1.340	1.300	1.281	1.267	1.263	0.251	0.126	0.084	0.031	0	0.064
Collab_2010	1.326	1.315	1.318	1.301	1.301	1.303	0.296	0.190	0.066	0.056	0.064	0

the answer to this question, we applied NetDistance on some real networks. With the aid of CiteSeerX digital library database [45], we extracted citation networks (named *Cit_CiteSeerX*) and collaboration networks (*Collab_CiteSeerX*) as the real networks data from the scientific publications. *Cit_CiteSeerX* shows the citations in the graph of the published papers, and *Collab_CiteSeerX* shows the co-authorships in the graph of authors. We extracted six snapshots from each network type from the year 2000 to 2010 biannually (2000, 2002, .. , 2010). For example, *Cit_2008* is the citation network snapshot of the papers published before 2008. The considered network snapshots show a diverse set of network sizes. The citation networks grow from about 800,000 nodes and 6,000,000 edges in *Cit_2000* to 1,000,000 nodes and 11,000,000 edges in *Cit_2010.* The size of the collaboration network extends from about 300,000 nodes and 1,000,000 edges in *Collab_2000* to 700,000 nodes and 3,000,000 edges in *Collab_2010.*

Although the structure of the networks evolves over time, a network snapshot is supposed to be similar to other snapshots of the same network. For example, *Cit_2010* is expected to be more similar to *Cit_2008* than to *Collab_2010.* Table 4 shows the pairwise distances for different citation and collaboration network snapshots, according to NetDistance measure. The table shows that NetDistance measurements are consistent with the expectation: NetDistance calculates the dissimilarities in a way that networks of the same type are considered more similar to each other than to networks of different types. Therefore, if we perform a KNN classification on the 12 network snapshots based on NetDistance dissimilarity function, the classifier correctly categorizes all the networks into two groups of citation and collaboration networks. Figure 7 shows a two-dimensional multidimensional scaling plot based on the reported distances of the network snapshots. Multidimensional scaling (MDS) roughly visualizes the level of similarity among the individuals in a dataset. The described experiments in this subsection show the capability of NetDistance for explaining the structural similarities in real network data.

6.2. Model Fitting for Real Networks

In this subsection, we apply the proposed model fitting algorithm on real networks. First, the citation and collaboration networks described in the previous subsection are used as the real networks data. For all of the citation network snapshots, the model selector returns the "Random Power-law" [4] as the selected model, and it suggests the "Forest Fire" model [6] for all of the collaboration network snapshots. Therefore, the performance of the proposed model selection method is consistent for different snapshots of the same network type. Additionally, we considered

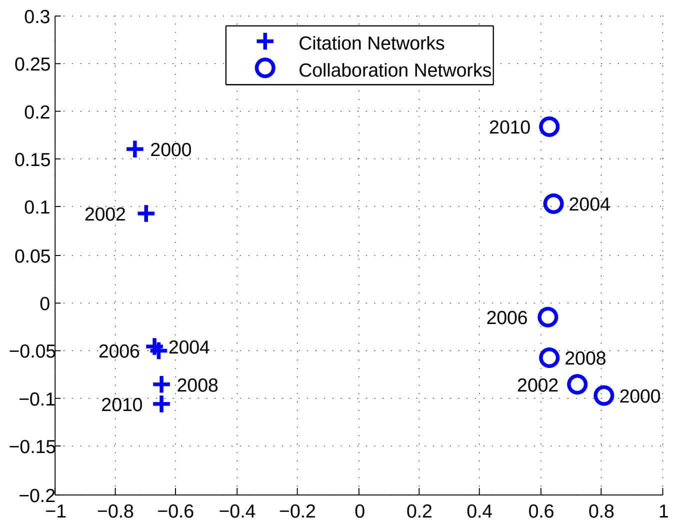


Figure 7: Multidimensional scaling plot for citation and collaboration network snapshots. The plot shows that NetDistance successfully categorizes the network snapshots into two meaningful groups.

two other real networks: SlashdotZoo as a social network and Gnutella as a peer-to-peer (P2P) network. SlashdotZoo is a friendship network dataset of users of the technology news site *Slashdot* (`http://slashdot.org`). SlashdotZoo is the snapshot of that social network in 2009, which consists of about 80,000 nodes and 500,000 edges. The Gnutella P2P network, which consists of about 6,000 nodes and 20,000 edges, is the graph of Gnutella hosts in 2002 [1]. Our model selection algorithm proposes "Kronecker Graphs" model [5] for Gnutella and "Barabási-Albert" model [2] for SlashdotZoo.

In this experiment, we compare real networks with their corresponding generated networks in order to show that given a real network, ModelFit is capable of generating artificial networks with similar structures (perhaps with different sizes). In this regard, ModelFit algorithm is applied on four real networks from different types, and the selected model and estimated parameters are determined for each real network. The examined networks are: Gnutella (P2P network), SlashdotZoo (social network), Collab_2010 (collaboration network), and Cit_2010 (citation network). Corresponding to each real network, an artificial network is generated using the selected model and estimated parameters. Each synthesized network consists of 10,000 nodes (except for GnutellaGen which is generated by Kronecker graphs model and thus contains $2^{13} = 8,192$ nodes). Table 5 shows the pairwise distances for real networks and the synthesized artificial networks. In this table, the "Gen" postfix is attached to the name of the generated networks. As the table reveals, the most similar network to each real network is its fitted artificial counterpart. In other words, ModelFit has resulted in generation of artificial networks, each of which is structurally similar to its corresponding original real network. This is while the sizes of the real networks are different from those of the corresponding artificial networks. The case study shows that given a target network or its topological features, ModelFit is capable of generating structurally similar networks, with a desired (smaller or larger) network size.

7. Computational Cost Analysis of the Proposed Method

In this section, we analyze the algorithm complexity of the proposed ModeFit method and its applicability for large networks. ModelFit consists of two components: Model selection and parameter estimation, both of which are developed based on supervised machine learning algorithms. As we described in previous chapters, the proposed method is

[1]SlashdotZoo and Gnutella datasets are available at public network dataset repositories, such as `http://snap.stanford.edu/data` and `http://konect.uni-koblenz.de/networks`. CiteSeerX snapshots are available upon request.

Table 5: Pairwise distances for real networks and the generated networks. The most similar network to each real network is its fitted artificial counterpart.

	Cit_2010	Cit_2010Gen	Collab_2010	Collab_2010Gen	Gnutella	GnutellaGen	SlashdotZoo	SlashdotZooGen
Cit_2010	0	0.578	1.303	0.971	0.801	1.007	0.638	0.650
Cit_2010Gen	0.578	0	1.656	1.476	0.917	1.096	0.960	0.712
Collab_2010	1.303	1.656	0	0.964	2.010	2.227	1.857	1.876
Collab_2010Gen	0.971	1.476	0.964	0	1.431	1.668	1.545	1.611
Gnutella	0.801	0.917	2.010	1.431	0	0.487	0.560	0.492
GnutellaGen	1.007	1.096	2.227	1.668	0.487	0	0.500	0.585
SlashdotZoo	0.638	0.960	1.857	1.545	0.560	0.500	0	0.249
SlashdotZooGen	0.650	0.712	1.876	1.611	0.492	0.585	0.249	0

learned based on a training set of artificial networks and the learned methods are applicable for input networks. When the learned ModeFit is applied on a target network, it first computes the network features, selects a fitting model, and finally estimates the model parameters for generating networks similar to the target network. Among these three steps, feature extraction is the main time-consuming part since it processes the (perhaps very large) input network and calculates its corresponding feature vector. The small fixed-size feature vector is then compared with a small set of feature vectors (less than 60 vectors in the neighbors set [2]) so that to find the nearest neighbor, and to select the best fitting model. Thus, the model selection is performed in a rather constant time ($O(1)$). The parameter estimation is also achieved by applying the extracted feature into a learned neural network. Since the neural network has a small number of processing nodes, the computational cost of the neural network execution is also constant ($O(1)$). Table 6 shows the time complexity of the utilized feature extraction algorithms. This table shows that the overall ModelFit feature extraction complexity is $O(E)$. In this table, E denotes the number of edges, and V denotes the number of nodes in the target network. In addition to the features mentioned in Table 6, we had also considered two other network features: Average shortest path length [33] and effective diameter [6]. When we applied machine learning methods (genetic algorithms and neural networks) for developing ModelFit, the weight of these two features is always assigned equal to zero. In other words these two features do not incorporate in final learned ModeFit method, and therefore they are not included in time complexity analysis. Since the model selection and parameter estimation algorithms are performed in $O(1)$ (in a constant time), the overall time complexity of ModelFit for model selection and parameter estimation of a target network is $O(E)$. In other words, the algorithm is performing linear to the number of the edges in the target network, and thus is scalable.

The "learning phase" of the proposed method is irrelevant to computational cost analysis of the proposed method since the learning is performed only once, and then the learned methods are applied for each input network data. Nevertheless, it is worth noting that the learning phase of both model selection and parameter estimation components in the proposed method follow efficient algorithms. The model selection is based on running a genetic algorithm in less than 200 iterations on a population of 540 individual chromosomes, which converges in less than an hour in an ordinary PC station with single processor and 4 gigabytes of RAM. Moreover, an artificial neural network is learned for model parameter estimation in less than 5 minutes. It is worth noting that the learning algorithms are applied on rather small networks with less than 5,000 nodes, and thus the feature extraction and the computations in the learning phase is fast and efficient.

8. Conclusion

In this paper, we investigated the problem of model fitting for complex networks, which includes the model selection and parameter estimation subproblems. In this application, we have a target network (or its feature values)

[2]Refer to the model selection algorithm for details

Table 6: Time complexity of ModelFit feature extraction components.

Network Feature	Time Complexity
Clustering Coefficient	$O(E)$
Transitivity	$O(E)$
Assortativity	$O(E)$
Modularity	$O(V)$
Degree Distribution (DDQC)	$O(E)$
Density	$O(E)$
Average Degree	$O(E)$
Total (Maximum)	$O(E)$

in hand and we seek answers for two consequent questions. First: Among the candidate network models, which one is most suitable for generating networks similar to the given target network? Second: What are the best parameter values for the selected model to generate networks similar to the target network? We showed that machine learning algorithms are effective in both of the problems. In the model selection problem, the distance-based methods outperform the methods that are based on supervised classification learning. Among the distance-based methods, the nearest neighbor classification outperformed the average-distance based methods. On the other hand, in parameter estimation problem, supervised learning of an artificial neural network results in a better parameter estimator than the distance-based (nearest neighbor) approach.

The proposed ModelFit method includes learning a distance function (NetDistance) to be used in nearest neighbor classification for model selection, along with an ANN per candidate network model, learned to estimate the generative parameters. The evaluations shows that ModelFit outperforms alternative approaches with respect to accuracy and noise tolerance. ModelFit can play an important role in many network applications including simulations, network sampling, extrapolation, anonymization, network summarization, and network comparison. The main contributions of this research can be concluded as the following:

1. Integration of various local/global features in network comparison, classification, and parameter estimation, which results in an accurate model fitting method.
2. Using distance based classification for predicting network models, which results in a noise-tolerant model selection method.
3. Favoring nearest neighbor classification over average distance based methods, which increases the accuracy of model selection.
4. Applying machine learning algorithms in order to develop a special-purpose distance function that is capable of separating networks of candidate models.
5. Comprehensive empirical evaluations based on different datasets, many network instances, and different noise levels.
6. Utilization of supervised machine learning algorithms (Artificial Neural Networks) to develop a parameter estimation method, with a methodology which is independent of the network models.

9. Acknowledgement

This research is supported by Iran Telecommunication Research Center (ITRC).

Appendix A. The Network Models

The utilized network generation models and the synthesized graphs of the artificial networks dataset are described in the following:
Erdős-Rényi (ER). This model generates completely random graphs with a specified density [40]. In our implementation, the density parameter is selected randomly from the range: $0.002 \leqslant density \leqslant 0.005$.
Barabási-Albert model (BA) [2]. In this model, a new node is randomly connected to k existing nodes with a probability that is proportional to the degree of the available nodes. The range of k parameter in our evaluations is $1 \leqslant k \leqslant 10$.

Watts-Strogatz model (WS). The classical Watts-Strogatz small-world model [3] starts with a regular lattice, in which each node is connected to k neighbors, and then randomly rewires some edges of the network with rewiring probability β. In our implementation, β is fixed as $\beta = 0.5$, and k is selected from the integer numbers between 2 and 10 ($2 \leqslant k \leqslant 10$).
Forest Fire (FF). This model [6] is configured by two main parameters: Forward burning probability (p) and backward burning probability (p_b). We fixed $p_b = 0.32$ and selected p from the range $0 \leqslant p \leqslant 0.3$.
Kronecker graphs (KG). This model is based on applying a matrix operation on a small initiator matrix [5]. We utilized 2×2 initiator matrices for generating networks of this model. The four elements of the initiator matrix are selected from the ranges: $0.7 \leqslant P_{1,1} \leqslant 0.9, 0.5 \leqslant P_{1,2} \leqslant 0.7, 0.4 \leqslant P_{2,1} \leqslant 0.6, 0.2 \leqslant P_{2,2} \leqslant 0.4$.
Random power-law (RP). This model [4] is configured by the power-law degree exponent (γ). In our parameter settings, γ is randomly selected from the range $2.5 < \gamma < 3$.

[1] R. Albert, A.-L. Barabási, Statistical mechanics of complex networks, Reviews of modern physics 74 (1) (2002) 47–97.
[2] A.-L. Barabási, R. Albert, Emergence of scaling in random networks, Science 286 (5439) (1999) 509–512.
[3] D. J. Watts, S. H. Strogatz, Collective dynamics of 'small-world' networks, nature 393 (6684) (1998) 440–442.
[4] D. Volchenkov, P. Blanchard, An algorithm generating random graphs with power law degree distributions, Physica A: Statistical Mechanics and its Applications 315 (3) (2002) 677–690.
[5] J. Leskovec, D. Chakrabarti, J. Kleinberg, C. Faloutsos, Z. Ghahramani, Kronecker graphs: An approach to modeling networks, The Journal of Machine Learning Research 11 (2010) 985–1042.
[6] J. Leskovec, J. Kleinberg, C. Faloutsos, 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, ACM, 2005, pp. 177–187.
[7] R. Pastor-Satorras, A. Vespignani, Epidemic dynamics in finite size scale-free networks, Physical Review E 65 (3) (2002) 035108.
[8] A. Montanari, A. Saberi, The spread of innovations in social networks, Proceedings of the National Academy of Sciences of the United States of America 107 (47) (2010) 20196–20201.
[9] L. Briesemeister, P. Lincoln, P. Porras, Epidemic profiles and defense of scale-free networks, in: Proceedings of the 2003 ACM workshop on Rapid malcode, ACM, 2003, pp. 67–75.
[10] K. M. Borgwardt, Graph kernels, Ph.D. thesis, Ludwig-Maximilians-Universität München (2007).
[11] A. Sala, L. Cao, C. Wilson, R. Zablit, H. Zheng, B. Y. Zhao, Measurement-calibrated graph models for social network experiments, in: Proceedings of the 19th international conference on World wide web, ACM, 2010, pp. 861–870.
[12] P. Mahadevan, D. Krioukov, K. Fall, A. Vahdat, Systematic topology analysis and generation using degree correlations, ACM SIGCOMM Computer Communication Review 36 (4) (2006) 135–146.
[13] E. M. Airoldi, X. Bai, K. M. Carley, Network sampling and classification: An investigation of network model representations, Decision support systems 51 (3) (2011) 506–518.
[14] M. P. Stumpf, C. Wiuf, Sampling properties of random graphs: the degree distribution, Physical Review E 72 (3) (2005) 036118.
[15] S. H. Lee, P.-J. Kim, H. Jeong, Statistical properties of sampled networks, Physical Review E 73 (1) (2006) 016102.
[16] M. P. Stumpf, C. Wiuf, R. M. May, Subnets of scale-free networks are not scale-free: sampling properties of networks, Proceedings of the National Academy of Sciences of the United States of America 102 (12) (2005) 4221–4224.
[17] J. Leskovec, C. Faloutsos, Sampling from large graphs, in: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2006, pp. 631–636.
[18] J.-D. J. Han, D. Dupuy, N. Bertin, M. E. Cusick, M. Vidal, Effect of sampling on topology predictions of protein-protein interaction networks, Nature biotechnology 23 (7) (2005) 839–844.
[19] S. Chester, B. M. Kapron, G. Srivastava, S. Venkatesh, Complexity of social network anonymization, Social Network Analysis and Mining 3 (2) (2013) 151–166.
[20] B. Zhou, J. Pei, W. Luk, A brief survey on anonymization techniques for privacy preserving publishing of social network data, ACM SIGKDD Explorations Newsletter 10 (2) (2008) 12–22.
[21] G. Wondracek, T. Holz, E. Kirda, C. Kruegel, A practical attack to de-anonymize social network users, in: Security and Privacy (SP), 2010 IEEE Symposium on, IEEE, 2010, pp. 223–238.
[22] S. Bhagat, G. Cormode, B. Krishnamurthy, D. Srivastava, Class-based graph anonymization for social network data, Proceedings of the VLDB Endowment 2 (1) (2009) 766–777.
[23] A. Kelmans, Comparison of graphs by their number of spanning trees, Discrete Mathematics 16 (3) (1976) 241–261.
[24] Ö. N. Yaveroğlu, N. Malod-Dognin, D. Davis, Z. Levnajic, V. Janjic, R. Karapandza, A. Stojmirovic, N. Pržulj, Revealing the hidden language of complex networks, Scientific reports 4.
[25] N. Pržulj, D. G. Corneil, I. Jurisica, Modeling interactome: scale-free or geometric?, Bioinformatics 20 (18) (2004) 3508–3515.
[26] N. Pržulj, Biological network comparison using graphlet degree distribution, Bioinformatics 23 (2) (2007) e177–e183.
[27] K. Faust, Comparing social networks: size, density, and local structure, Metodološki zvezki 3 (2) (2006) 185–216.
[28] S. Motallebi, S. Aliakbary, J. Habibi, Generative model selection using a scalable and size-independent complex network classifier, Chaos: An Interdisciplinary Journal of Nonlinear Science 23 (4) (2013) 043127.
[29] J. Janssen, M. Hurshman, N. Kalyaniwalla, Model selection for social networks using graphlets, Internet Mathematics 8 (4) (2012) 338–363.
[30] M. Middendorf, E. Ziv, C. H. Wiggins, Inferring network mechanisms: the drosophila melanogaster protein interaction network, Proceedings of the National Academy of Sciences of the United States of America 102 (9) (2005) 3192–3197.
[31] O. Kuchaiev, A. Stevanović, W. Hayes, N. Pržulj, Graphcrunch 2: software tool for network modeling, alignment and clustering, BMC bioinformatics 12 (2011) 24.
[32] N. Shervashidze, T. Petri, K. Mehlhorn, K. M. Borgwardt, S. Viswanathan, Efficient graphlet kernels for large graph comparison, in: Interna-

tional Conference on Artificial Intelligence and Statistics, 2009, pp. 488–495.

[33] L. d. F. Costa, F. A. Rodrigues, G. Travieso, P. Villas Boas, Characterization of complex networks: A survey of measurements, Advances in Physics 56 (1) (2007) 167–242.

[34] L. Hamill, N. Gilbert, Social circles: A simple structure for agent-based social network models, Journal of Artificial Societies and Social Simulation 12 (2) (2009) 3.

[35] M. E. Newman, Modularity and community structure in networks, Proceedings of the National Academy of Sciences of the United States of America 103 (23) (2006) 8577–8582.

[36] M. E. Newman, Assortative mixing in networks, Physical review letters 89 (20) (2002) 208701.

[37] S. Aliakbary, J. Habibi, A. Movaghar, Quantification and comparison of degree distributions in complex networks, in: Proceedings of The Seventh International Symposium on Telecommunications (IST2014), IEEE, 2014, to appear.

[38] S. Aliakbary, J. Habibi, A. Movaghar, Quantification and comparison of network degree distributions, arXiv preprint arXiv:1307.3625.

[39] D. E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, 1st Edition, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989.

[40] P. Erdös, A. Rényi, On the central limit theorem for samples from a finite population, Publications of the Mathematical Institute of the Hungarian Academy 4 (1959) 49–61.

[41] S. Kotsiantis, Supervised machine learning: A review of classification techniques, Informatica 31 (2007) 249–268.

[42] J. C. Platt, Fast training of support vector machines using sequential minimal optimization, in: Advances in kernel methods, MIT press, 1999, pp. 185–208.

[43] L. Akoglu, C. Faloutsos, Rtg: a recursive realistic graph generator using random typing, Data Mining and Knowledge Discovery 19 (2) (2009) 194–209.

[44] K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators, Neural networks 2 (5) (1989) 359–366.

[45] Citeseerx digital library, `http://citeseerx.ist.psu.edu`.
URL `http://citeseerx.ist.psu.edu`