Robust Parameter Fitting to Realistic Network Models via Iterative Stochastic Approximation

Thomas Bläsius
Karlsruhe Institute of Technology
Karlsruhe, Germany
thomas.blaesius@kit.edu

Sarel Cohen
The Academic College of Tel Aviv-Yaffo
Tel Aviv, Israel
sarelco@mta.ac.il

Philipp Fischbeck
Tobias Friedrich
philipp.fischbeck@hpi.de
tobias.friedrich@hpi.de
Hasso Plattner Institute, University of Potsdam
Potsdam, Germany

Martin S. Krejca
LIX, CNRS, Ecole Polytechnique, Institut Polytechnique de Paris
Palaiseau, France
martin.krejca@polytechnique.edu

ABSTRACT

Random graph models are widely used to understand network properties and graph algorithms. Key to such analyses are the different parameters of each model, which affect various network features, such as its size, clustering, or degree distribution. The exact effect of the parameters on these features is not well understood, mainly because we lack tools to thoroughly investigate this relation. Moreover, the parameters cannot be considered in isolation, as changing one affects multiple features. Existing approaches for finding the best model parameters of desired features, such as a grid search or estimating the parameter–feature relations, are not well suited, as they are inaccurate or computationally expensive.

We introduce an efficient iterative fitting method, named *ParFit*, that finds parameters using only a few network samples, based on the Robbins-Monro algorithm. We test ParFit on three well-known graph models, namely Erdős–Rényi, Chung–Lu, and geometric inhomogeneous random graphs, as well as on real-world networks, including web networks. We find that ParFit performs well in terms of quality and running time across most parameter configurations.

CCS CONCEPTS

• **Theory of computation** → *Graph algorithms analysis*; **Random network models**; **Stochastic approximation**.

KEYWORDS

Random Network Models, Stochastic Approximation, Network Features, Parameters

1 INTRODUCTION

In the current age of big data, a lot of important information exists as huge networks, such as interaction networks, semantic networks, and social networks. Their immense size makes certain tasks very challenging, for example, sharing, processing, or reasoning about these networks. A common way to overcome this challenge is to utilize *models*, which aim to describe networks via a small amount of parameters. Ideally, these model parameters are sufficient to capture all of the important features of real-world networks, thus reducing their complexity and allowing to handle tasks more easily.

A fundamental concept for this purpose are random graph models [15]. Given a certain set of parameters, such models do not construct a specific graph but instead a random graph out of a family where most samples are likely to exhibit similar features. This is oftentimes beneficial, as it, for example, allows to share the structure of a network without sharing the exact network, or to produce different benchmarks with similar properties. Due to this importance, various random graph models have been analyzed with respect to how well they can reproduce real-world graphs, such as exponential random graph models [3], Kronecker graphs [18, 26], approaches based on clustering [5, 21], approaches based on embeddings [9, 19], and further models [20, 27, 33]. In order for each model to reproduce an input well, it is essential to choose the parameters of the model carefully. This poses a challenging task and strongly depends on the network models, since the relationship between input parameters and output features might not always be clear.

To this end, suitable parameters are selected in various ways, depending on what information about the model is available. If applicable, gradient descent [26] or moment-based methods [18] are good approaches for finding the log-likelihood maximizer of the parameters. If no such closed expression is known, an alternative is to estimate parameters based on domain knowledge (e.g., an estimated linear relation, monotonic behavior, or a formula) [7, 29]. In case of a lot of data, techniques from machine learning to train a network distance function can be used in order to choose nearest neighbors of possible parameter configurations [2]. If (almost) no information is available, running a grid search and choosing the best parameters based on a distance measure [28, 33] is a possibility.

Although these approaches can yield good results, they come with their own limitations. Minimizing the gradient typically results in hard, non-convex optimization problems [15]. Similarly, the moment-method usually requires a huge amount of samples, rendering it not very efficient [18]. Domain knowledge needs to be specific to be of use. For example, the temperature parameter of the geometric inhomogeneous random graph model is known to be negatively correlated to the expected clustering coefficient of the generated networks, but the exact relation is unclear and depends on other model parameters. A linear estimation, as suggested in [23], is not very accurate. Techniques from machine learning

usually yield intransparent weight functions and involve a computationally expensive training phase. Last, a grid search for the best parameters is computationally expensive and has limited accuracy.

Another challenge all approaches have to face is that many random graph models do not guarantee to construct connected graphs, whereas real-world networks in available datasets are often connected. This is usually not due to the represented real-world network being completely connected but due to the dataset being only its largest connected component, which can originate in the dataset creation method (repeated neighbor selection) or during some preprocessing, since usually only the largest component is of interest. Determining the largest component of the artificial graph resulting from models might affect some of its features, most notably its number of vertices and its average degree. Thus, the relation between the input parameters and output features becomes more complex. While there is still a correlation, the exact relation is not known. Previous works mainly deal with models generating connected networks [28], or estimate the effect on the number of vertices, but not on the other parameters, thus applying the reduction to the largest component after parameter fitting [7].

We propose a method, named *ParFit*, that is designed to circumvent these problems. ParFit fits the parameters of a collection of random network models while requiring only few network model samples to reach well fitting model parameters. This method involves no training phase and can deal with the complex parameter landscape introduced by reducing to the largest component.

1.1 Setting

We consider three well-known random graph models, namely, Erdős–Rényi, Chung–Lu with power-law degree distribution, and geometric inhomogeneous random graphs (GIRGs). For all of these models, we consider the variant of the model where only the largest component is returned. For every model, we provide a correspondence of model parameters to measurable network features with a strong correlation; however, changing one parameter can change multiple features, and the effect after only considering the largest component is unclear. Further, due to the random nature of each model, we only have access to samples from its probability distribution. Under these constraints, given a model and a graph, the goal is to find model parameters that, in expectation, yield networks with the corresponding features matching those of the given graph.

While there are asymptotic results on the giant components and connectivity of the models based on the parameters [10–13, 16, 17] as well as results on predicting features for the model without reduction to the largest component [10, 11], to the best of our knowledge, no formulas for expected values of these models' parameters and features we consider are known, in particular for the GIRG model with reduction to the largest component. This eliminates the likelihood- and moments-based methods mentioned above.

1.2 Our Method: ParFit

We treat our setting as a root-finding problem and present a parameter fitting method for this scenario (*ParFit*, Section 4) based on the Robbins-Monro method [31] in stochastic approximation. In an iterative process, a model parameter guess is established, and based on a single model network sample with these parameters, the parameters are updated according to the deviation of the sample feature values from the target features values. To account for fluctuations around the optimal parameter values, the final parameters are the mean over the parameters of the most recent iterations.

1.3 Contribution

We show the effectiveness of ParFit by evaluating it on a wide range of scenarios, including both random graph models as well as real-world networks. For all three random graph models, we find that ParFit yields parameters that fit very well within only few iterations (Table 1 and Figure 1), even in the difficult regime of low vertex degree, where reducing to the largest component has a large influence on all network features. For real-world networks (Section 6), we also observe that ParFit works well, including the low-degree regime. Especially, our fitted power-law exponents are similar to those estimated in related work. Overall, ParFit is effective and only needs few iterations to find suitable model parameters.

2 PROBLEM DEFINITION

Given a network model M as well as an undirected, unweighted graph G, we aim to find parameters for M that provide the best fit for G with respect to a well chosen metric. While the best metric would be to compare the parameters of M and G, this is generally not possible, as the parameters of are not directly observable in G. Hence, we choose our metric based on the following observation: For every parameter, there is usually a corresponding *feature* that is measurable in any graph, which is mainly controlled by this parameter. Our goal is to find model parameters such that these measurable features match the features of G in expectation.

To formalize this, we call M a *random graph model* with p parameters $\theta = (\theta_1, \dots, \theta_p) \in \mathbb{R}^p$ if $M(\theta)$ is a probability distribution over the set of all graphs $\mathcal{G}$, i.e., the outcome of each trial is a graph and $M(\theta)$ assigns a probability to each $G \in \mathcal{G}$. We write $H \sim M(\theta)$ to indicate that $H \in \mathcal{G}$ was sampled from $M(\theta)$. To improve readability, we add θ as a subscript to the sampled graph, i.e., $H_\theta \sim M(\theta)$ reminds the reader that H_θ was sampled using the parameters θ.

A model can be equipped with *measurable features* $\varphi_1, \dots, \varphi_p$, where each measurable feature φ_i for $i \in [p]$ is a function mapping a graph to a numerical value, i.e., $\varphi_i \colon \mathcal{G} \to \mathbb{R}$. We assume a one-to-one correspondence between the parameters of the model and the measurable features and say that φ_i is the measurable feature *corresponding* to the parameter θ_i for $i \in [p]$. A measurable feature φ_i is a good choice if it can be efficiently evaluated, if φ_i is strongly correlated with its corresponding parameter θ_i, and if it has only a minor dependence on other parameters θ_j with $j \neq i$. For brevity, we also write $\varphi(G) = (\varphi_1(G), \dots, \varphi_p(G))$ for a graph $G \in \mathcal{G}$.

With this formalization, we define the *parameter fitting problem*. Given a random graph model M equipped with measurable features φ and given a graph G, find parameter values θ such that for a graph $H_\theta \sim M(\theta)$ sampled from M, the expected measurable features are close to the features of G, i.e., $\|\mathrm{E}[\varphi(H_\theta)] - \varphi(G)\|$ is minimized.

This is essentially a multivariate root-finding problem (when assuming there is a solution with error 0), with the difficulty that the measurable features of sampled graphs might have high variance. Thus, we can view this as a stochastic optimization problem with noisy evaluations. Moreover, note that the input graph G is not

really used except for evaluating the measurable features. Thus, one can also view $\varphi(G)$ as being the input instead of G itself.

3 NETWORK MODELS

We consider several random graph models. Here, we discuss their parameters and our choice of corresponding measurable features.

Erdős–Rényi Graphs. Given two parameters n and k, the Erdős–Rényi model [16] generates a graph with n vertices where any pair of vertices is connected with probability $p = k/(n-1)$ independently of all other choices. We consider the model variant that reduces the resulting graph to its largest connected component.

As corresponding measurable features, we use the *number of vertices* of a graph to correspond to the parameter n and the *average degree* to correspond to k. These are canonical choices, as the initially generated graph has n vertices and expected average degree k. Note, however, that only considering the largest connected component makes this connection less direct.

Chung–Lu Graphs. In the Chung–Lu model [1, 11, 12], edges are drawn independently with varying probabilities, allowing for a heterogeneous degree distribution. Every vertex has a weight, and every pair of vertices is connected with probability proportional to the product of their weights. The expected degree of each vertex then roughly follows its weight. We consider a power-law version of this model where the weight distribution follows a power law with exponent β. Formally, the model has three parameters n, k and β. We use n vertices, with vertex i having weight $w_i = c \cdot i^{-1/(\beta-1)}$, where c is such that the average weight is k, i.e., the total weight is $W = kn$. Then, vertices u and v are adjacent with probability

$$p(u,v) = \min(1, w_u w_v / W).$$

We reduce the resulting graph to its largest connected component.

As before, we use the *number of vertices* of a graph as measurable feature corresponding to the parameter n and the *average degree* as measurable feature corresponding to k. The power-law exponent β controls the variance of the degree distribution, with lower β yielding a higher variance. A way of measuring this is the *heterogeneity* as defined by Bläsius and Fischbeck [6], which is the base 10 logarithm of the coefficient of variation of a graph's degree distribution. As the heterogeneity is negatively correlated with β, we use its negation as measurable feature corresponding to β.

Geometric Inhomogeneous Random Graphs. The model of geometric inhomogeneous random graphs (GIRGs) [10] is similar to the Chung–Lu model but adds dependencies between edges, using an underlying geometry. We use a 1-dimensional geometry for which the GIRG model is closely related to hyperbolic random graphs [23].

As in the Chung–Lu model, we have the parameters n, k, and β with similar meanings as before; see details below. Additionally, there is the *temperature* T as fourth parameter, controlling the strength of the geometry and thereby the amount of dependence between the edges. Given these parameters, a GIRG is generated as follows. Start with n vertices and assign each vertex v a random weight w_v following a power-law distribution with exponent β and a random position x_v drawn uniformly from the interval $[0,1]$. The distance between two vertices u and v with positions x_u and x_v is $\|x_u - x_v\| = \min(|x_u - x_v|, 1 - |x_u - x_v|)$. With this, any pair of vertices u and v have an edge between them with probability

$$p(u,v) = \min\left(1, c\big(w_u w_v / (\|x_u - x_v\| W)\big)^{1/T}\right),$$

where W is the sum of all weights and the constant c is chosen such that the resulting graph has expected average degree k [8]. Finally, the resulting graph is reduced to its largest connected component.

Like in the Chung–Lu model, as measurable features, we use the *number of vertices* corresponding to the parameter n, the *average degree* corresponding to k, and the negative *heterogeneity* [6] corresponding to β. The temperature T affects the influence of the geometry. A stronger geometry leads to the emergence of more triangles, and a common way to measure the amount of triangles is the *average local clustering coefficient* (*clustering coefficient* for short). The clustering coefficient is the probability that a random vertex together with two random neighbors form a triangle. As the clustering coefficient is negatively correlated with the temperature T, we use its negation as measurable feature corresponding to T.

Algorithm 1: Our parameter fitting method *ParFit* that returns for a given random graph model M, measurable features φ and target values $\varphi(G)$ a set of parameters θ such that M parameterized by θ exhibits features close to $\varphi(G)$. See Section 4 for more details.

Input: Model M, features φ, target feature values $\varphi(G)$
Output: Model parameters

$\theta_0 \leftarrow$ initial configuration;
$i_\pm \leftarrow 30$;
for $i \leftarrow 0$ **to** ∞ **do**
  $H_{\theta_i} \leftarrow$ sample from M_{θ_i};
  $\Delta_i \leftarrow \varphi(G) - \varphi(H_{\theta_i})$;
  $\theta_{i+1} \leftarrow \theta_i + \Delta_i$;
  if sign change in all features at least once **then**
    $i_\pm \leftarrow \min(i_\pm, i)$;
  if $i \geq i_\pm$ **then**
    $\overline{\theta}_i \leftarrow \frac{1}{i - i_\pm + 1} \sum_{j=i_\pm}^{k} \theta_j$;
    if $i \geq i_\pm + 200$ **or** $\overline{\theta}_i$ converged **then return** $\overline{\theta}_i$;

4 OUR PARAMETER FITTING METHOD

Our method *ParFit* (see Algorithm 1) is an iterative stochastic approximation method. It is based on the *Robbins-Monro* algorithm [31], originally introduced to solve a root-finding problem by exploiting the monotonicity of the root function and modifying the result based on whether the current guess is too high or too low. Similarly, ParFit maintains a choice of parameter values for the parameter fitting problem and iteratively adjusts each parameter randomly based on its corresponding measurable feature. This results in an anytime algorithm, i.e., it can be interrupted at any time to output its current solution. Thus, when to stop the algorithm is a trade-off between solution quality and run time.

As the standard Robbins-Monro algorithm leads to oscillations around optima, we improve the convergence behavior of ParFit by adding *iterate averaging* [30, 32]. That is, starting from some fixed iteration, we log all subsequent results, and once ParFit is

stopped, it returns the arithmetic mean of all logged values as the final solution. We choose this iteration during the run, based on a extension of an adaptive approach [22] to the multivariate case [14].

Algorithm Description. Given a random graph model M with measurable features φ as well as target values $\varphi(G)$, ParFit starts with an initial parameter choice θ_0 (discussed below). In each iteration i, we compute a single sample $H_{\theta_i} \sim M(\theta_i)$ of the model M with parameters θ_i. Then we compute the new solution θ_{i+1} from θ_i by adjusting each input parameter proportional to the deviation $\Delta_i := \varphi(G) - \varphi(H_{\theta_i})$ of the corresponding feature of the sampled graph from the target value $\varphi(G)$. In step i, we scale Δ_i by a weight a_i, called the *gain*. The dependence of a_i on i makes it possible to, e.g., introduce a cool-down by gradually reducing a_i, which can prevent oscillation. Formally, we set $\theta_{i+1} = \theta_i + a_i\Delta_i$.

For every feature, we keep track of the earliest iteration in which the sign of Δ_i is different than that of Δ_{i+1}. We set $i_\pm$ to be the earliest iteration in which this has happened for all features, and we store solutions from that point on.

When the algorithm is terminated in iteration i^*, we return the arithmetic mean of all stored values, i.e., $\overline{\theta}_{i^*} := \frac{1}{i^*-i_\pm+1}\sum_{j=i_\pm}^{i^*}\theta_j$.

Parameter Choices. A common choice for a_i is $(i+1)^{-\alpha}$ for some non-negative α. However, in many contexts, $\alpha = 0$ and thus $a_i = 1$ is chosen [24, 34], even though this does not guarantee convergence [34]. In our scenario, this choice yields a good fit (Section 5.3).

For all models, we configure the algorithm as follows. For the initial solution θ_0, we take the number of vertices n and the average degree to be those of the target values $\varphi(G)$, and choose an initial temperature of 0.5 and an initial power-law exponent of 3.0. Further, we set $i_\pm$ to be at most 30. Last, we terminate the procedure once the final solution[1] $\overline{\theta}_i$ fulfills the following convergence criterion: For the last 10 iterations, the relative change of $\overline{\theta}_i$ across all parameters is below 1 %; these thresholds are further analyzed in Section 5.3. We set a maximum of 200 such averaging iterations to ensure termination; however, this maximum was never reached in our experiments for our choice of algorithm configuration.

Remarks. While in general, one might have to add a factor to Δ_i to account for imbalanced scaling between parameters and features, this was not necessary for our parameters and features.

We compute θ_{i+1} via only a single sample H_{θ_i} from the model, which gives a very coarse estimation of the expected feature values $E[\varphi(H_{\theta_i})]$. However, this is not a problem as repeated iterations mitigate these noisy evaluations. Moreover, the iterate averaging helps reduce the effect of outliers when close to the optimum.

5 EVALUATION

We evaluate how well ParFit (Algorithm 1) is able to recover the model parameters of random graph models. For an evaluation on real-world networks, please refer to Section 6.

We consider a *predictive simulation* in which we fit a model to given networks, take samples based on the fitted parameters, and compare the features of the samples to those of the original networks. In order to only measure the quality of ParFit and not of the model, we choose only networks that actually come from the same network model, ensuring that the fitting is actually possible. We consider different quality measures and discuss the results.

[1] That is, the arithmetic mean of all so-far stored solutions.

5.1 Setup

We implemented ParFit in Python, using several libraries. For network property analysis as well as the Erdős–Rényi and Chung–Lu model, we utilize the networKit library [4]. For the GIRG model, we employ the efficient generator by Bläsius et al. [8]. The experiments were run on a Macbook Pro with an Apple M1 chip and 16 GB RAM. All code and data is published at https://github.com/PFischbeck/parameter-fitting-experiments.

We consider a range of parameter configurations for all network models. In particular, for the GIRG model, we choose number of vertices $n = 10\,000$, average degree ranging from 2 to 10, power-law exponent ranging from 2.1 to 25.0, and temperature ranging from 0.01 to 0.9999. For the Erdős–Rényi model and Chung–Lu model, we let the number of vertices range between 1000 and 10 000. In total, there are 171, 500 and 500 parameter configurations for the Erdős–Rényi, Chung–Lu and GIRG model respectively.

Recall that ParFit aims to minimize the difference between the target feature values and the expected actual feature values at the fitted parameters. We aim to measure the quality of ParFit with respect to predicting parameters for scenarios that are actually achievable with the given model, in expectation; we focus on the quality for individual networks in Section 6. To ensure the target feature values are achievable in expectation, for every considered parameter configuration, we take 50 samples from the model, and consider the mean corresponding features across the samples as the input $\varphi(G)$ for ParFit. For every input, we run ParFit and take 50 samples based on the fitted parameters. We measure the features of those samples and compare their mean to the feature values given to ParFit. While our problem definition states that we consider the vector length of those differences, it is more useful to look at each feature individually. For every feature, we measure the Pearson correlation coefficient as well as the mean absolute error, i.e., the mean of the absolute difference between the value of the target feature value and the mean of the 50 samples across all settings.

5.2 Results & Discussion

Table 1 provides an overview of the measured qualities of ParFit across the different models and scenarios. Across all features, we see a very strong Pearson correlation. Considering the number of vertices, the mean absolute error is very low for the Erdős–Rényi model as well as the Chung–Lu model. It is slightly increased for the GIRG model. Note however that all initial parameter configurations for the GIRG model were chosen such that $n = 10\,000$, while this parameter varies for the other models. The 90th percentile of the absolute error for GIRGs is at 113.2, indicating the effect of some outliers, which we consider closer in a moment. The average degree has a very low mean absolute error across all models. The other two features of heterogeneity and clustering are strongly correlated too. Recalling that the heterogeneity ranges roughly from -0.5 to 0.75, the MAEs of 0.01 and 0.02 are very low. The clustering feature values can range from 0 to 1, but the MAE for the GIRG model is only at 0.004. The mean iteration count is highest for the GIRG

Table 1: For the three considered models and the range of settings described in the setup (Section 5.1), we provide statistics on the quality of ParFit (Algorithm 1). For the four considered features (number of vertices, average degree, heterogeneity and clustering), we consider the mean actual feature values in relation to the target feature values, and provide the Pearson correlation coefficient as well as the mean absolute error (MAE). Values are left blank if not applicable for the respective models. In addition, we provide the mean number of iterations. Please refer to Section 5.2 for a discussion.

Model	Number of vertices		Average degree		Heterogeneity		Clustering		Iterations
	Pearson	MAE	Pearson	MAE	Pearson	MAE	Pearson	MAE	
Erdős–Rényi	0.999	2.6	0.999	0.02					13.6
Chung–Lu	0.999	6.5	0.999	0.01	0.999	0.01			23.3
GIRG	0.999	51.1	0.999	0.02	0.998	0.02	0.999	0.004	32.2

Table 2: For the GIRG model and varying α values for the ParFit step size, we provide statistics on ParFit's quality. For the four considered features (number of vertices, average degree, heterogeneity, clustering), we consider mean actual feature values in relation to target feature values, and provide the mean absolute error (MAE). We also provide the mean iteration count. Please refer to Section 5.3 for a discussion.

α	Mean absolute error				Iterations
	Vertices	Avg. deg.	Het.	Clu.	
0.0	50.6	0.02	0.02	0.004	32.0
0.2	42.7	0.02	0.02	0.004	29.0
0.4	46.9	0.02	0.03	0.004	28.3
0.6	63.5	0.03	0.04	0.006	28.2
0.8	96.5	0.04	0.05	0.007	30.3
1.0	126.9	0.05	0.06	0.009	32.2

Table 3: For the GIRG model and varying relative threshold values *(Thrsh.)* for the convergence stopping criterion, we provide statistics on the quality of ParFit. For the four considered features (number of vertices, average degree, heterogeneity and clustering), we consider mean actual feature values in relation to target feature values, and provide the mean absolute error (MAE). In addition, we provide the mean iteration count. Please refer to Section 5.3 for a discussion.

Thrsh.	Mean absolute error				Iterations
	Vertices	Avg. deg.	Het.	Clu.	
0.001	49.9	0.02	0.01	0.003	104.9
0.005	48.2	0.02	0.02	0.003	44.0
0.010	50.0	0.02	0.02	0.004	32.4
0.050	61.1	0.03	0.02	0.004	21.8
0.100	61.1	0.02	0.02	0.004	20.9

model; however, this can be expected as it has the most parameters and corresponding features out of the considered models.

Figure 1 shows the target and mean features of the fitted samples for the GIRG model. Across all four features (heterogeneity, clustering, average degree, number of vertices), the values of the samples from the fitted parameters closely follow the values given to ParFit. Even in those scenarios we consider very difficult, for example those with low average degree leading to a target vertex count considerably below 10 000, ParFit manages to find suitable parameters. We see that the number of iterations is usually highest for those scenarios with extreme target values, i.e., high clustering coefficient and low heterogeneity. The cases of low vertex count are not inherently hard because of this feature value, but rather the target vertex count is very low if the average degree is small.

There are some scenarios where ParFit seems to not give perfectly fitting parameters. In particular, for very low heterogeneity, ParFit finds parameters that are slightly too high. In addition, for high heterogeneity, fitted parameters yield graphs with heterogeneity slightly too high or low. For the clustering, there are some parameter configurations in the high clustering regime where ParFit struggles to yield matching parameters. And for the number of vertices, there are several scenarios where the number of vertices of the fitted samples is slightly higher than required. We can also see that those are the cases where the number of iterations taken by ParFit is high. Overall, there are some outliers where ParFit yielded parameters that do not provide perfectly fitting features. However, these cases are rare and coincide with a high number of iterations.

In Section A (Appendix), we provide the corresponding figures for the Erdős–Rényi and the Chung–Lu power-law model. In these cases, the parameters fit very well again. In the case of the Erdős–Rényi model in particular, low-degree scenarios lead to a higher number of iterations, due to the increased influence of the reduction to the largest component. However, ParFit deals with these cases easily, finding suitable parameters in under 20 iterations.

For the Chung–Lu power-law model, the method behavior is similar to that of the GIRG model scenario. In particular, the number of vertices and the average degree are mostly tightly fitted, except for few outliers where the method also takes substantially more iterations. For the heterogeneity, we can see that for scenarios with low heterogeneity, the method usually yields parameters that lead to a slightly too high heterogeneity, and it takes up to 60 iterations in such cases. Overall, ParFit provides fitting parameters across a wide variety of scenarios for all three considered models.

5.3 Algorithm Configuration

Since ParFit can be configured (Section 4), we discuss our configuration choice and evaluate the effect of varying its parameters.

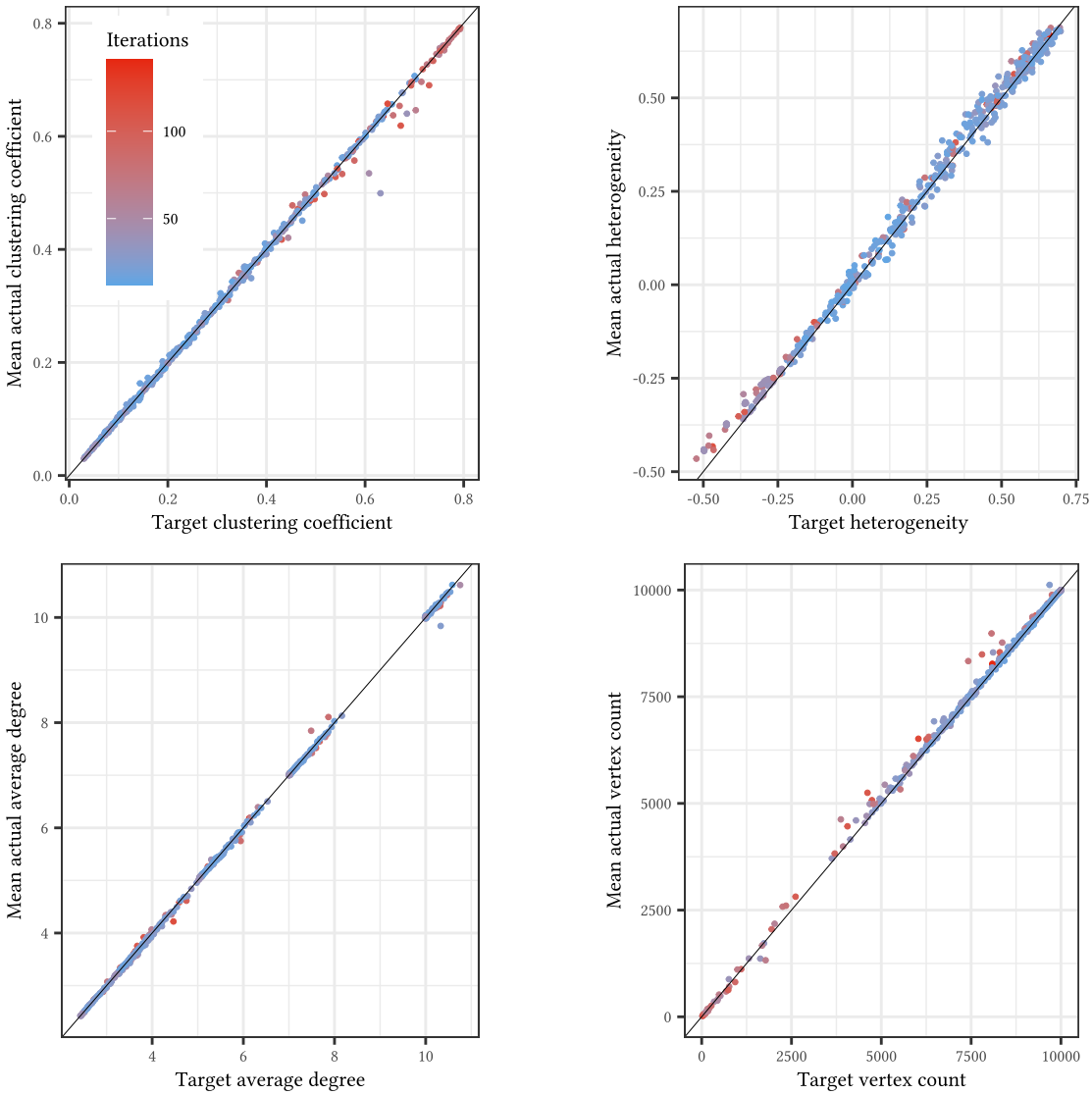

Figure 1: ParFit (Algorithm 1) evaluated on several scenarios of geometric inhomogeneous random graphs (GIRG; see also Section 3) instances. We sample 50 GIRG instances from a parameter configuration, take their mean corresponding features, run ParFit, and take 50 samples based on the fitted parameters. For the four relevant features of GIRG instances, the plots show the target feature value (given to ParFit; x-axis) as well as the mean actual feature value of 50 samples based on the fitted parameters (y-axis). The color shows the number of iterations (i.e., number of samples) taken by ParFit. A darker color indicates fewer iterations. The diagonal line shows the identity function. Please refer to Section 5.2 for a discussion.

Non-constant Gain. One aspect of ParFit is the choice of the gain function for the step size. We chose a constant gain of 1; however, in literature, a gain of $(k+1)^{\alpha}$ is commonly used [34], where k is the iteration, and α is a non-negative constant.

We consider a range of values for α ranging from 0 to 1 and how this affects the number of iterations as well as the quality of the fitted parameters, across the same scenarios described above. In particular, for the same models and scenarios, we run the altered ParFit version and again measure the resulting features.

Table 2 shows the resulting mean absolute error (MAE) values for the GIRG model when considering α values of 0.0, 0.2, 0.4, 0.6, 0.8, and 1.0. See Appendix A for results on other models. We measure only little change in the number of iterations across all models. However, the MAE across the four considered features increases with increasing α, supporting our choice of $\alpha = 0$. The exception is a slightly higher MAE for the number of vertices for low α. In contrast, the MAE of the heterogeneity on the GIRG model increases from 0.022 to 0.055 when going from $\alpha = 0$ to $\alpha = 1$. The interaction

of higher α with iterate averaging was nicely explained by Spall [34]: With decreased gain, the process gets closer to the optimal parameters from one side only, while iterate averaging performs best when the process moves around these optimal parameters.

Sign Change Phase. In our algorithm, we configure $i_{\pm}$ to be the earliest iteration in which all features had a sign change at some point, but we set it to at most 30 to ensure termination. In the experiments discussed in Section 5.2, this limit was reached for 160 out of the 1171 algorithm runs. In particular, reaching the limit was most commonly due to no sign changes in the heterogeneity in the low heterogeneity regime. In these scenarios, ParFit struggled to reach such low heterogeneity; see Figure 1.

Convergence Criterion. In our method, we stop the algorithm once the relative change in the current solution was below the threshold 1 % across all parameters for the last 10 iterations. We consider different values for this threshold to observe the effect on number of iterations and solution quality. Note that we still limit the number of averaging iterations to at most 200. This limit was only reached for 107 out of 5855 algorithm runs.

Table 3 shows the resulting mean absolute errors (MAE) for the GIRG model (see Appendix A for results on other models). As expected, a higher (i.e., more tolerant) threshold for convergence leads to a decreased number of iterations but also to slightly higher MAE values. Since we still enforce at least 10 iterations, even the solutions where the method is stopped early seem to provide generally good values. Our final choice of 0.01 for the threshold provides a good middle ground between few iterations and low MAE values.

6 APPLICATIONS

We apply ParFit (Algorithm 1) with the GIRG model to 35 undirected real-world networks from the KONECT database [25]. Table 4 shows the results. The expected values for the measurable features were obtained by sampling 50 graphs for the fitted parameters.

Quality of Fit. We first discuss the number of vertices and the average degree. For many networks, the average degree is sufficiently high such that the model generates a connected graph. In these cases, simply setting n to the number of vertices yields the correct result. Similarly, setting the parameter k to the desired average degree works well as the fitter for the average degree of the GIRG generator is already very good [8]. Although ParFit does not know this, it easily finds the correct values for n and k in these cases.

In the more interesting cases of low average degrees, ParFit works very well. For all but two networks, the deviation between the desired and the mean number of vertices is below 1 %. For the other two, `Youtube` and `Hyves`, the error is 1.87 % and 2.04 %. For the average degree, we have a similar picture, with deviations below 1 %, except for `Youtube` and `Hyves` with 2.47 % and 2.78 %. To highlight one example where ParFit provides new capabilities that were not available before, consider the `Route views` network. It has 6.5 k vertices and an average degree of 3.9. To generate a connected graph of that size and density with the same clustering and heterogeneity using the GIRG model, it is necessary to generate a graph with 13.4 k vertices and average degree 1.94, i.e., twice the number of vertices and half the average degree.

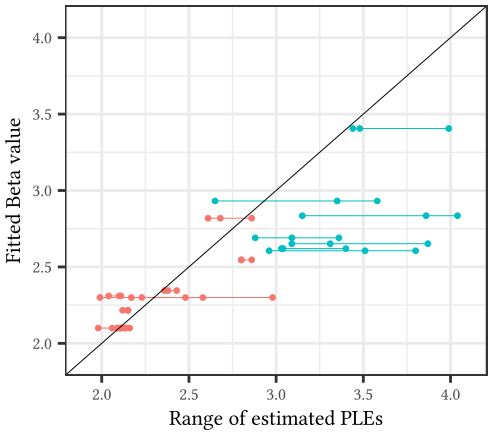

Figure 2: The power law exponents (PLEs) of all real-world networks (x-axis) versus the respective PLEs fitted by ParFit (Algorithm 1, y-axis) assuming a GIRG model (Section 3). The x-axis shows the range of the three PLE estimators in [35]. Red lines refer to strong power laws, blue to the rest. The diagonal is the identity function. See Section 6 for a discussion.

The two other measurable features are less directly controllable by model parameters than the number of vertices and the average degree. In fact, for some real-world networks, the measurable features might take values that are impossible to produce with the GIRG model. For the clustering coefficient, the largest deviation between desired and achieved value is only 0.06. For the heterogeneity, the deviation is also small for most instances. However, there are some instances with very high or very low heterogeneity, where the deviation is a bit higher. This is particularly true for the four road networks (`Roads TX/CA/PA` and `Chicago roads`). We want to stress that this is not a flaw of ParFit but rather indicates that the model is not a good representation of these types of networks. Although GIRGs can be used to generate homogeneous graphs by setting the power-law exponent very high (which is equivalent to giving all vertices the same weight), the degree of each vertex still follows a binomial distribution, which has some amount of variance. This makes it impossible to generate GIRGs that have the same heterogeneity as regular graphs (or graphs that are too close to being regular). Detailed figures for the heterogeneity and clustering coefficient can be found in Appendix B.

Temperature and Power-Law Exponent. Observe that the fitted temperatures are rather high for most networks. We offer three potential explanations for this. First, the influence of a latent underlying geometry is rather low. Second, the one-dimensional geometry of the model does not fully capture the higher-dimensional nature of the networks. Increasing the dimension has a similar effect on the clustering coefficient as increasing the temperature. And third, using the clustering coefficient as a measure for the strength of the underlying geometry oversimplifies matters. An example for this are the road networks, where it is reasonable to assume that the influence of an underlying geometry is rather high. However, the considered road networks have a low clustering coefficient and

Table 4: Comparison of four different features (*Actual*; number of vertices, average degree, heterogeneity, clustering) of real-world networks *(Graph)* to the average of those features of 50 samples from ParFit (Algorithm 1) fitted to each network *(Measured)*, using the GIRG model (Section 3). The column following each feature shows the respective GIRG model parameter from ParFit. The last column is the number of iterations that ParFit ran before terminating. See Section 6 for a discussion.

Graph	Number of Vertices		n	Average degree		k	Heterogeneity		β	Clustering		T	Iterations
	Actual	Measured		Actual	Measured		Actual	Measured		Actual	Measured		
CAIDA (IN)	26 475	26 463	51 917	4.0	4.1	2.1	0.92	0.80	2.10	0.21	0.21	0.79	22
Skitter (SK)	1 694 616	1 705 244	1 716 088	13.1	13.0	12.9	1.02	1.10	2.35	0.26	0.24	0.86	25
Actor collaborations (CL)	374 511	374 511	374 511	80.2	80.3	80.2	0.31	0.33	2.87	0.78	0.78	0.16	21
Amazon (CA)	334 863	334 867	342 937	5.5	5.5	5.4	0.02	0.01	3.41	0.40	0.40	0.59	18
arXiv (AP)	17 903	17 903	17 903	22.0	22.0	22.0	0.15	0.18	2.98	0.63	0.63	0.42	15
Bible names (MN)	1707	1710	1717	10.6	10.6	10.5	0.23	0.25	2.69	0.71	0.71	0.32	18
Brightkite (BK)	56 739	56 792	57 758	7.5	7.5	7.4	0.44	0.41	2.61	0.17	0.17	0.87	18
Catster (Sc)	148 826	148 826	148 826	73.2	73.2	73.2	1.05	0.79	2.10	0.39	0.38	0.92	16
Catster/Dogster (Scd)	601 213	601 213	601 214	52.1	52.1	52.1	0.96	0.94	2.31	0.50	0.48	0.71	24
Chicago roads (CR)	12 979	12 992	13 872	3.2	3.2	3.0	−0.45	−0.29	10.11	0.04	0.04	0.99	40
DBLP (CD)	317 080	319 189	334 410	6.6	6.6	6.4	0.18	0.18	3.10	0.63	0.63	0.36	19
Dogster (Sd)	426 485	426 486	426 488	40.1	40.1	40.1	0.85	0.86	2.22	0.17	0.19	0.98	17
Douban (DB)	154 908	155 044	174 190	4.2	4.2	3.8	0.44	0.41	2.56	0.02	0.04	1.00	17
U. Rovira I Virgili (A@)	1133	1133	1135	9.6	9.6	9.6	−0.01	0.00	2.94	0.22	0.22	0.86	18
Euro roads (ET)	1039	1043	1310	2.5	2.5	2.1	−0.32	−0.26	6.17	0.02	0.05	1.00	42
Flickr (LF)	1 624 991	1 625 669	1 626 051	19.0	19.0	19.0	0.84	0.84	2.47	0.19	0.16	0.89	20
Flickr (FI)	105 722	105 722	105 722	43.8	43.8	43.8	0.42	0.40	2.49	0.09	0.11	1.00	26
Flixster (FX)	2 523 386	2 539 464	2 649 795	6.3	6.2	6.0	0.77	0.76	2.53	0.08	0.08	0.94	17
Gowalla (GW)	196 591	197 260	198 829	9.7	9.6	9.6	0.74	0.57	2.55	0.24	0.21	0.83	20
Hamsterster (Shf)	1788	1787	1789	14.0	14.0	14.0	0.18	0.17	2.46	0.14	0.20	1.00	18
Hamsterster (Sh)	2000	2000	2000	16.1	16.2	16.1	0.12	0.12	2.90	0.54	0.55	0.52	19
Hyves (HY)	1 402 673	1 431 249	1 983 390	4.0	3.9	2.8	1.06	1.06	2.30	0.04	0.05	0.99	32
LiveJournal (Lj)	5 189 808	5 189 824	5 189 986	18.8	18.8	18.8	0.43	0.46	2.84	0.27	0.27	0.75	24
Livemocha (LM)	104 103	104 103	104 103	42.1	42.1	42.1	0.42	0.39	2.50	0.05	0.11	1.00	15
Orkut (OR)	3 072 441	3 072 441	3 072 441	76.3	76.3	76.3	0.31	0.28	2.93	0.17	0.16	0.84	17
Power grid (UG)	4941	4942	6356	2.7	2.7	2.2	−0.17	−0.16	3.85	0.08	0.08	0.88	19
Proteins (Mp)	1458	1448	2303	2.7	2.7	1.8	0.11	0.12	2.65	0.07	0.07	0.96	19
Reactome (RC)	5973	5973	5973	48.8	48.8	48.8	0.14	0.16	2.85	0.61	0.61	0.46	17
Roads CA (RO)	1 957 027	1 952 063	2 203 991	2.8	2.8	2.5	−0.45	−0.28	10.52	0.05	0.05	0.91	40
Roads PA (RD)	1 087 562	1 089 165	1 222 095	2.8	2.8	2.6	−0.45	−0.28	10.37	0.05	0.05	0.91	40
Roads TX (R1)	1 351 137	1 355 658	1 535 134	2.8	2.8	2.5	−0.44	−0.28	10.16	0.05	0.05	0.91	40
Route views (AS)	6474	6499	13 432	3.9	3.9	1.9	0.81	0.64	2.10	0.25	0.25	0.70	24
WordNet (WO)	145 145	145 224	146 237	9.0	9.0	9.0	0.36	0.35	2.82	0.60	0.60	0.45	26
Youtube (CY)	1 134 890	1 156 102	1 339 791	5.3	5.1	4.5	0.98	1.01	2.30	0.08	0.07	0.98	26
Human PPI (MV)	2783	2790	3033	4.3	4.3	4.0	0.21	0.17	2.62	0.07	0.09	1.00	22

thus require high temperature. This is due to connections being often subdivided, i.e., there are many vertices of degree 2, which heavily decreases the number of triangles. We believe that all three explanations have merit for different networks and that it is an interesting future question to study which explanation is the right one for which network. We note that, though this is beyond the scope of this paper, ParFit enables the study of these kind of questions by running experiments on models with higher dimensions or other measurable features corresponding to the temperature.

The values we obtain for the power-law exponents are not surprising. Figure 2 shows them in comparison to the results of Voitalov et al. [35], who applied three different estimation methods. The figure includes all networks that were classified as power-law networks [35]. In most cases, our obtained power-law exponents are within the range obtained by Voitalov et al. or close to it.

This is insofar interesting, as the two approaches have different objectives. The goal of Voitalov et al. [35] is to find an exponent such that the observed tail-distribution best fits a power-law distribution with that exponent, potentially taking cutoffs into account. ParFit on the other side aims at finding a power-law distribution whose variance best fits the variance observed in the degree distribution. Both approaches lead to similar results, indicating the heterogeneity measure we use is well suited as a proxy for the power-law exponent. Additionally, differences between the resulting power-law exponents may in part come from the fact that we reduced the networks to their largest connected component, while Voitalov et al. [35] considered the whole network's distribution.

7 CONCLUSION

We have presented a fitting method (ParFit; Algorithm 1) that, given an Erdős–Rényi, Chung–Lu or GIRG model as well target feature values, finds model parameters that lead to the desired features on average. We have shown that ParFit works well for a wide range of scenarios, assuming the target feature values are achievable by the model. We have applied ParFit to real-world networks covering several areas, including infrastructure and online social networks. In these cases, ParFit still provides well-fitting parameters and closely matches related work on the scale-freeness of these networks.

We think our work is applicable to other random graph models, including hyperbolic random graphs. Our work now allows for the use of accurate parameter fitting in many contexts, e.g., further fitting the models to real-world networks in order to properly measure how realistic other features of the sampled model instances are. Further improvements to the fitting method are also possible, e.g., a better approximation of the gain via the Kiefer–Wolfowitz algorithm in the simultaneous perturbation variant.

REFERENCES

[1] William Aiello, Fan Chung, and Linyuan Lu. 2000. A Random Graph Model for Massive Graphs. In *STOC*, Vol. 2000. Citeseer, 1–10.

[2] Sadegh Aliakbary, Sadegh Motallebi, Sina Rashidian, Jafar Habibi, and Ali Movaghar. 2015. Noise-Tolerant Model Selection and Parameter Estimation for Complex Networks. *Physica A: Statistical Mechanics and its Applications* 427 (2015), 100–112.

[3] Weihua An. 2016. Fitting ERGMs on Big Networks. *Social Science Research* 59 (2016), 107–119.

[4] Eugenio Angriman, Alexander van der Grinten, Michael Hamann, Henning Meyerhenke, and Manuel Penschuck. 2022. Algorithms for Large-Scale Network Analysis and the NetworKit Toolkit. In *Algorithms for Big Data: DFG Priority Program 1736*. Springer Nature Switzerland, 3–20. https://doi.org/10.1007/978-3-031-21534-6_1

[5] Shweta Bansal, Shashank Khandelwal, and Lauren A. Meyers. 2009. Exploring Biological Network Structure with Clustered Random Networks. *BMC Bioinformatics* 10, 405 (2009). https://doi.org/10.1186/1471-2105-10-405

[6] Thomas Bläsius and Philipp Fischbeck. 2022. On the External Validity of Average-Case Analyses of Graph Algorithms. In *30th Annual European Symposium on Algorithms (ESA 2022) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 244)*, Shiri Chechik, Gonzalo Navarro, Eva Rotenberg, and Grzegorz Herman (Eds.). Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 21:1–21:14. https://doi.org/10.4230/LIPIcs.ESA.2022.21

[7] Thomas Bläsius, Tobias Friedrich, Maximilian Katzmann, Anton Krohmer, and Jonathan Striebel. 2018. Towards a Systematic Evaluation of Generative Network Models. In *International Workshop on Algorithms and Models for the Web-Graph*. Springer, 99–114.

[8] Thomas Bläsius, Tobias Friedrich, Maximilian Katzmann, Ulrich Meyer, Manuel Penschuck, and Christopher Weyand. 2022. Efficiently Generating Geometric Inhomogeneous and Hyperbolic Random Graphs. *Network Science* 10, 4 (2022), 361–380.

[9] Marián Boguñá, Fragkiskos Papadopoulos, and Dmitri Krioukov. 2010. Sustaining the Internet with Hyperbolic Mapping. *Nature Communications* 1, 62 (2010).

[10] Karl Bringmann, Ralph Keusch, and Johannes Lengler. 2019. Geometric Inhomogeneous Random Graphs. *Theoretical Computer Science* 760 (2019), 35–54. https://doi.org/10.1016/j.tcs.2018.08.014

[11] Fan Chung and Linyuan Lu. 2002. The Average Distances in Random Graphs with given Expected Degrees. *Proceedings of the National Academy of Sciences* 99, 25 (2002), 15879–15882. https://doi.org/10.1073/pnas.252631999

[12] Fan Chung and Linyuan Lu. 2002. Connected Components in Random Graphs with given Expected Degree Sequences. *Annals of Combinatorics* 6, 2 (2002), 125–145. https://doi.org/10.1007/PL00012580

[13] Fan Chung and Linyuan Lu. 2006. The Volume of the Giant Component of a Random Graph with Given Expected Degrees. *SIAM Journal on Discrete Mathematics* 20, 2 (Jan. 2006), 395–411. https://doi.org/10.1137/050630106

[14] Bernard Delyon and Anatoli Juditsky. 1993. Accelerated Stochastic Approximation. *SIAM Journal on Optimization* 3, 4 (1993), 868–881.

[15] Mikhail Drobyshevskiy and Denis Turdakov. 2019. Random Graph Modeling: A Survey of the Concepts. *Comput. Surveys* 52, 6 (2019), 1–36.

[16] Paul Erdős and Alfréd Rényi. 1959. On Random Graphs I. *Publicationes Mathematicae* 6 (1959), 290–297.

[17] Paul Erdős and Alfréd Rényi. 1960. On the Evolution of Random Graphs. 5, 1 (1960), 17–60.

[18] David F. Gleich and Art B. Owen. 2012. Moment-Based Estimation of Stochastic Kronecker Graph Parameters. *Internet Mathematics* 8, 3 (2012), 232–256.

[19] Palash Goyal and Emilio Ferrara. 2018. Graph Embedding Techniques, Applications, and Performance: A Survey. *Knowledge-Based Systems* 151 (2018), 78–94.

[20] Alexander Gutfraind, Ilya Safro, and Lauren Ancel Meyers. 2015. Multiscale Network Generation. In *International Conference on Information Fusion*. IEEE, 158–165.

[21] Mark S. Handcock, Adrian E. Raftery, and Jeremy M. Tantrum. 2007. Model-Based Clustering for Social Networks. *Journal of the Royal Statistical Society Series A: Statistics in Society* 170, 2 (2007), 301–354.

[22] Harry Kesten. 1958. Accelerated Stochastic Approximation. *The Annals of Mathematical Statistics* 29, 1 (1958), 41–59.

[23] Dmitri Krioukov, Fragkiskos Papadopoulos, Maksim Kitsak, Amin Vahdat, and Marián Boguñá. 2010. Hyperbolic Geometry of Complex Networks. *Physical Review E: Statistical Physics, Plasmas, Fluids, and Related Interdisciplinary Topics* 82, 3 (2010), 036106. https://doi.org/10.1103/PhysRevE.82.036106

[24] C-M Kuan and Kurt Hornik. 1991. Convergence of Learning Algorithms with Constant Learning Rates. *IEEE Transactions on Neural Networks* 2, 5 (1991), 484–489.

[25] Jérôme Kunegis. 2013. KONECT – The Koblenz Network Collection. In *Proc. Int. Conf. on World Wide Web Companion*. 1343–1350. https://doi.org/10.1145/2487788.2488173

[26] Jure Leskovec, Deepayan Chakrabarti, Jon Kleinberg, Christos Faloutsos, and Zoubin Ghahramani. 2010. Kronecker Graphs: An Approach to Modeling Networks. *Journal of Machine Learning Research* 11 (2010), 985–1042.

[27] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. 2005. 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, Chicago Illinois USA, 177–187. https://doi.org/10.1145/1081870.1081893

[28] Marcell Nagy and Roland Molontay. 2019. On the Structural Properties of Social Networks and Their Measurement-Calibrated Synthetic Counterparts. In *Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*. 584–588.

[29] Marcell Nagy and Roland Molontay. 2022. Network Classification Based Structural Analysis of Real Networks and Their Model-Generated Counterparts. arXiv:1810.08498 [cs.SI]

[30] Boris T Polyak and Anatoli B Juditsky. 1992. Acceleration of Stochastic Approximation by Averaging. *SIAM journal on control and optimization* 30, 4 (1992), 838–855.

[31] Herbert Robbins, Sutton Monro, et al. 1951. A Stochastic Approximation Method. *The Annals of Mathematical Statistics* 22, 3 (1951), 400–407.

[32] D Ruppert. 1991. Stochastic Approximation. Handbook of Sequential Analysis, BK Ghosh and PK Sen, Eds.

[33] Alessandra Sala, Lili Cao, Christo Wilson, Robert Zablit, Haitao Zheng, and Ben Y. Zhao. 2010. Measurement-Calibrated Graph Models for Social Network Experiments. In *Proceedings of the 19th International Conference on World Wide Web (WWW '10)*. 861–870. https://doi.org/10.1145/1772690.1772778

[34] James C Spall. 2005. *Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control*. John Wiley & Sons.

[35] Ivan Voitalov, Pim Van Der Hoorn, Remco Van Der Hofstad, and Dmitri Krioukov. 2019. Scale-Free Networks Well Done. *Physical Review Research* 1, 3 (2019), 033034.

Table 7: For the Chung–Lu model and varying values for the convergence threshold, we provide statistics on the quality of ParFit. For the three considered features (number of vertices, average degree, heterogeneity), we consider the mean actual feature values in relation to the target feature values, and provide the mean absolute error (MAE). In addition, we provide the mean number of iterations.

Threshold	Mean absolute error			Iterations
	Vertices	Avg. deg.	Het.	
0.001	7.6	0.01	0.00	54.5
0.005	6.1	0.01	0.01	24.4
0.010	6.7	0.01	0.01	23.3
0.050	6.7	0.01	0.01	22.7
0.100	6.6	0.01	0.01	22.8

Table 8: For the Erdős–Rényi model and varying values for the convergence threshold, we provide statistics on the number on the quality of ParFit. For the two considered features (number of vertices, average degree), we consider the mean actual feature values in relation to the target feature values, and provide the mean absolute error (MAE). In addition, we provide the mean number of iterations.

Threshold	Mean absolute error		Iterations
	Vertices	Avg. deg.	
0.001	2.5	0.01	25.8
0.005	3.2	0.01	14.5
0.010	2.9	0.01	13.6
0.050	3.5	0.01	13.3
0.100	2.8	0.02	13.4

Table 5: For the Chung–Lu model and varying values of α for the ParFit step size, we provide statistics on the quality of ParFit. For the three considered features (number of vertices, average degree, heterogeneity), we consider the mean actual feature values in relation to the target feature values, and provide the mean absolute error (MAE). In addition, we provide the mean number of iterations.

α	Mean absolute error			Iterations
	Vertices	Avg. deg.	Het.	
0.0	6.8	0.01	0.01	23.2
0.2	5.8	0.01	0.01	24.9
0.4	6.7	0.01	0.02	27.5
0.6	10.5	0.01	0.02	31.4
0.8	21.2	0.02	0.03	36.1
1.0	44.3	0.03	0.05	37.7

Table 6: For the Erdős–Rényi model and varying values of α for the ParFit step size, we provide statistics on the quality of ParFit. For the two considered features (number of vertices, average degree), we consider the mean actual feature values in relation to the target feature values, and provide the mean absolute error (MAE). In addition, we provide the mean number of iterations.

α	Mean absolute error		Iterations
	Vertices	Avg. deg.	
0.0	2.8	0.01	13.4
0.2	2.6	0.01	13.6
0.4	2.5	0.01	13.8
0.6	3.2	0.01	14.2
0.8	4.4	0.02	14.8
1.0	7.4	0.02	15.6

A ADDITIONAL EVALUATION RESULTS

In this section, we provide further figures for the Erdős–Rényi model and Chung–Lu power-law model that did not fit into the main paper.

Figure 3 and Figure 4 show the differences between the target features and the mean of the fitted samples, similar to those for the GIRG model shown in Figure 1.

Table 5 and Table 6 show the quality measures for varying values of ParFit gain α, similar to those for the GIRG model shown in Table 2. In a similar fashion, Table 7 and Table 8 show the quality measures for varying values for the convergence threshold, similar to those for the GIRG model shown in Table 3. Across all tables, we see very similar results in that a low α is a good choice, and a medium threshold is a good trade-off between solution quality and iteration count.

B REAL-WORLD NETWORKS

Figure 5 shows information of the fitting method on the real-world networks discussed in Section 6 on the GIRG model; in particular on the accuracy in terms of heterogeneity and clustering coefficient.

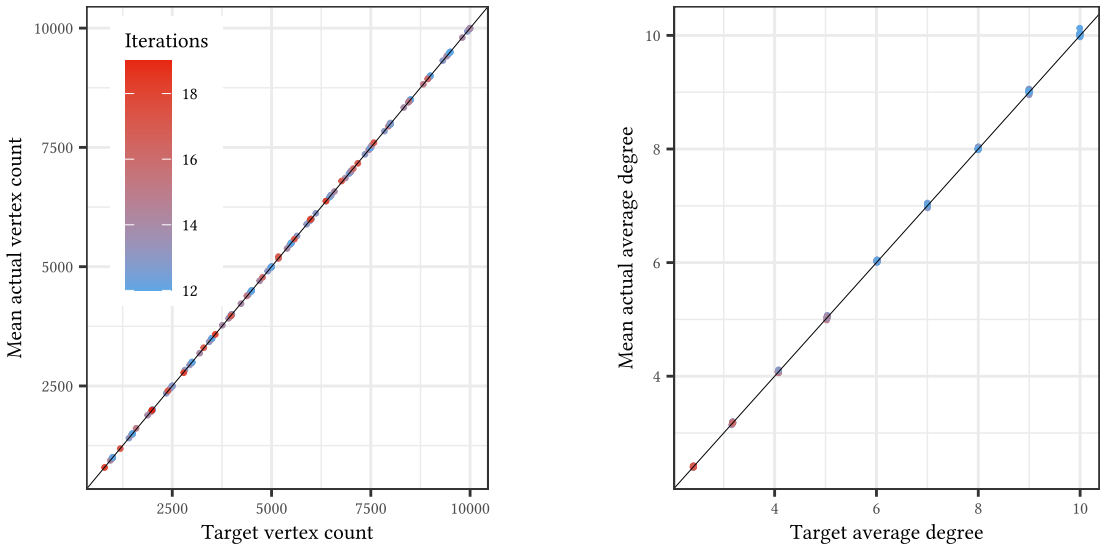

Figure 3: ParFit (Algorithm 1) evaluated on several Erdős–Rényi scenarios. We sample 50 Erdős–Rényi instances from a parameter configuration, take their mean corresponding features, run the parameter fitting algorithm, and take 50 samples based on the fitted parameters. For the two relevant features of Erdős–Rényi instances, the plots show the target feature value (given to ParFit; x-axis) as well as the mean actual feature value of 50 samples based on the fitted parameters (y-axis). The color shows the number of iterations (i.e., number of samples) taken by ParFit. A darker color indicates fewer iterations. The diagonal line shows the identity function.

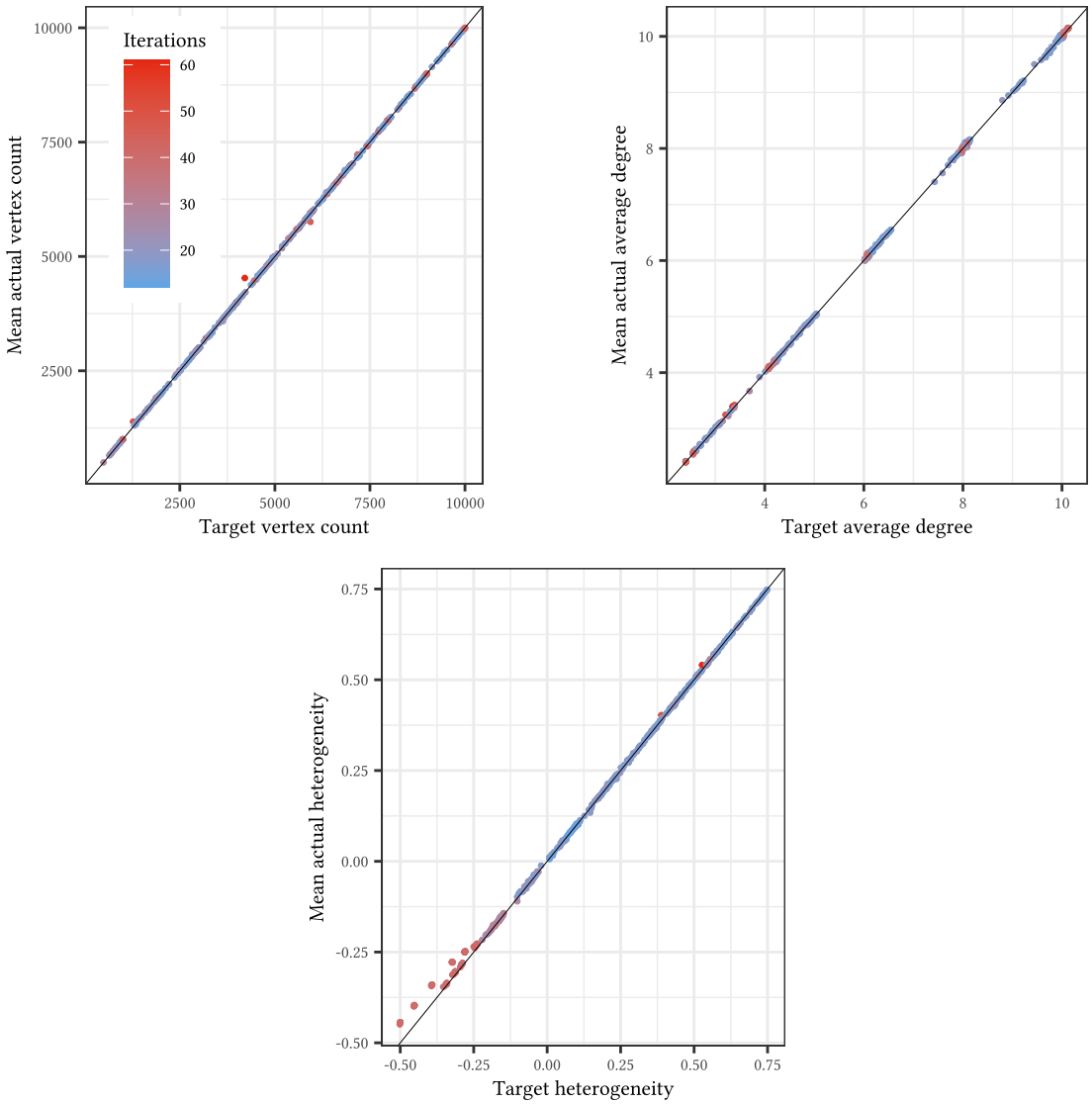

Figure 4: ParFit (Algorithm 1) evaluated on several Chung–Lu scenarios. We sample 50 Chung–Lu instances from a parameter configuration, take their mean feature values, run the parameter fitting algorithm, and take 50 samples based on the fitted parameters. For the three relevant features of Chung–Lu instances, the plots show the target feature value (given to ParFit; x-axis) as well as the mean actual feature value of 50 samples based on the fitted parameters (y-axis). The color shows the number of iterations (i.e., number of samples) taken by ParFit. A darker color indicates fewer iterations. The diagonal line shows the identity function.

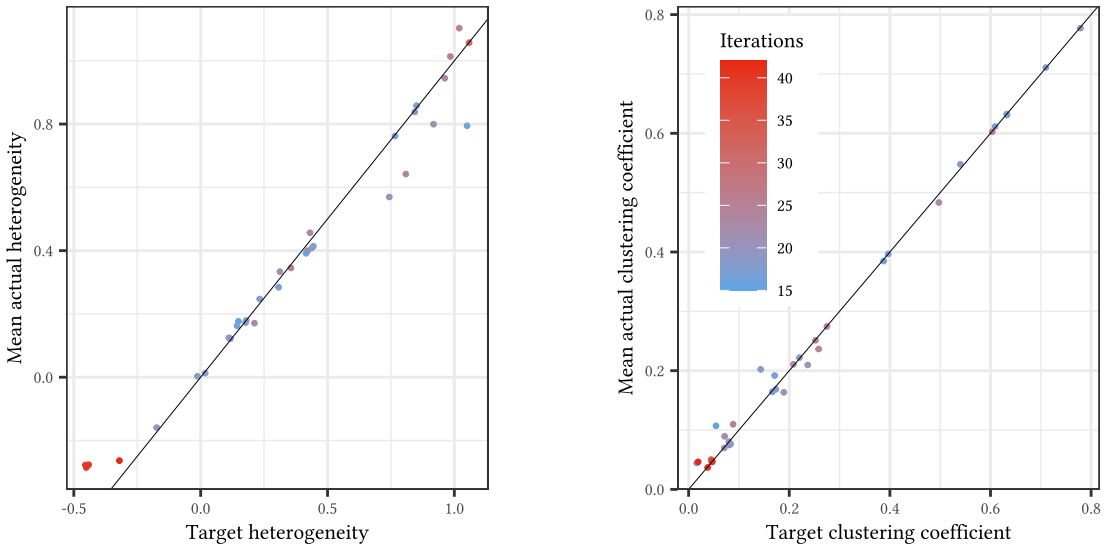

Figure 5: For every considered real-world network, we run the parameter fitting algorithm for the GIRG model on it, and take 50 samples based on the fitted parameters. The plot shows the true measured heterogeneity/clustering of the real-world network versus the mean heterogeneity/clustering of 50 samples based on the fitted parameters. The color shows the number of iterations (i.e., number of samples) taken by the fitting method. The diagonal line shows an identity function.