ORNL/TM-2016/3

Survey of Approaches to Generate Realistic Synthetic Graphs

Seung-Hwan Lim
Sangkeun Matt Lee
Sarah Powers
Mallikarjun Shankar
Neena Imam

October 2015

OAK RIDGE NATIONAL LABORATORY
MANAGED BY UT-BATTELLE FOR THE US DEPARTMENT OF ENERGY

ORNL/TM-2016/3

Computational Sciences and Engineering Division[⋆]
Computer Science and Mathematics Division[‡]
Computing and Computational Sciences Directorate[†]

Survey of Approaches to Generate Realistic Synthetic Graphs

Seung-Hwan Lim[⋆], Sangkeun Matt Lee[⋆], Sarah Powers[‡], Mallikarjun Shankar[⋆], Neena Imam[†]

October 2015

Prepared by
OAK RIDGE NATIONAL LABORATORY
P.O. Box 2008
Oak Ridge, Tennessee 37831-6285
managed by
UT-Battelle, LLC
for the
US DEPARTMENT OF ENERGY
under contract DE-AC05-00OR22725

CONTENTS

Abstract

A graph is a flexible data structure that can represent relationships between entities. As with other data analysis tasks, the use of realistic graphs is critical to obtaining valid research results. Using the actual ("real-world") graphs for research and new algorithm development is often difficult due to the presence of sensitive information in the data or due to the scale of data. This results in practitioners developing algorithms and systems that employ synthetic graphs instead of real-world graphs. Generating realistic synthetic graphs that provide reliable statistical confidence to algorithmic analysis and system evaluation involves addressing technical hurdles in a broad set of areas. This report surveys the state of the art in approaches to generate realistic graphs that are derived from fitted graph models on real-world graphs.

1. INTRODUCTION

A graph is a data structure that represents a set of entities as vertices where pairs of vertices are connected by edges that represent the relationships between vertices. Using graphs we can represent a variety of real-world information in a natural and efficient way. Structures such as social networks [40], Internet network topology [17], links between web pages [27], chemical structures [11], and genomic/proteomic information [10, 47] may be intuitively thought of as graph structures. Analyzing these graphs unveils useful implied knowledge such as recurring patterns and causal relationships. Performing the analysis on real-world graphs is critical to obtaining valid results for newly developed algorithms and system-implementation evaluations. Unfortunately, using the real-world graphs themselves for analysis is difficult due to the following two main reasons:

- Critical information: the owner of the information in the graphs may not want to share his/her data with specific groups of people (e.g., a third-party research entity or software developer) because the data may contain critical proprietary or protected information [24, 26, 46, 53].
- The scale of data: the scale of the empirical graph derived from real-world data can be inappropriate: it may either be too large or too small for practical use. Large-scale graphs incur large computational costs to share and use and are thus not ideal for the design and testing of new methods and techniques [34]. On the other hand, if we can only use a portion of the real-world data such as a subset of the communication network topology from the entire Internet [37] or a subset of user information from social network service providers [45], then the graphs these data sets lead to may be too small to be representative graphs for analysis. In these situations we have to "scale the data" in one or both directions: either we sub-sample the data rigorously from the large data set, or we extrapolate and create a large graph. However, in both directions, we would like to preserve similar statistical (in the aggregate sense) characteristics, and maintain the necessary structural fidelity of the real-world graph [30].

Hence, many research efforts use *realistic* synthetic graphs as an alternative to real-world graphs. This report surveys the state of art research for generating realistic synthetic graphs, along with associated research challenges and opportunities.

GENERATION MODELS AND APPROACH

A synthetic graph is a graph generated using a “graph generative model.” A realistic synthetic graph is a synthetic graph that uses fitted model parameters derived from empirical or real-world graphs. (We use the terms real-world graphs and empirical graphs to denote actual representations of the existing physical reality.) In order to provide reasonable statistical confidence in experimental results on the synthetic graphs, it is important to ensure that the generated graphs have properties that effectively mirror those found in the real-world graphs. Selecting a compact statistical model to match real-world data is a long-standing research area [35]. As a special case of statistical model construction, determining graph generative model parameters involves similar challenges in selecting model parameters to match both aggregate statistics while offering statistically valid realism. This realism is tested in the fidelity to real-world structures and sub-structures in the graph neighborhoods one or more edges away. Evaluating the quality of the generated synthetic graphs, and the pattern of access to the working set of data is a post-generation evaluation step.

Figure 1 outlines the approach for generating a realistic synthetic graph. The process of generating realistic synthetic graphs can be grouped into three tasks: *graph generative model selection*, *synthetic graph generation*, and *graph generative model validation*. We briefly discuss these steps.

1. Graph generative model selection: a graph generation model denotes a specific instance of a model

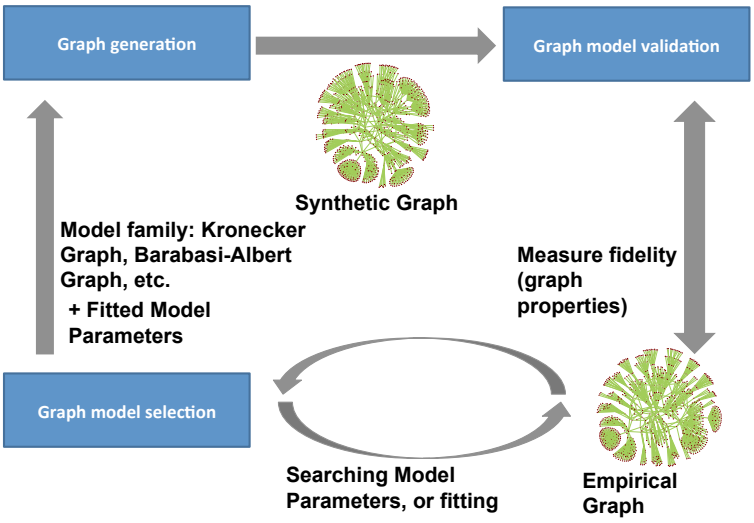

Fig. 1. The overview of three major tasks in generating realistic synthetic graphs: graph generative model selection, synthetic graph generation, and graph generative model validation.

family with model parameters, where a model family refers to a type of graph generation model. Well known models include Barabasi-Albert [8], Erdös-Rényi [16], and Kronecker graph generation models [30]. This step aims to determine a proper *graph generation model*, which includes a set of fitted model parameters derived from empirical real-world graphs. For this process, the major operation is to search the model parameters with a desired goodness of fit in accordance with a certain criteria. For searching model parameters, we might manually choose a family of models. For instance, we might choose either the Kronecker graph model or Nearest Neighbors model, and then search model parameters for the family. Or, we might consider multiple model families at the same time, while individually searching model parameters for models from each family. We often refer to this process as graph fitting, which can be an iterative process as with selecting a statistical model in general [54].

2. Synthetic graph generation: this is the task of producing synthetic graphs for a given graph generation model and a set of model parameters [12]. The computational challenges in generating synthetic graphs vary depending on the choice of model family. However, this process is typically a memory or I/O-bound operation in order to manage the data structure for storing graphs [36].

3. Graph generation model validation: understanding the discrepancy between real-world graphs and synthetic graphs is critical, as it will dictate the statistical confidence of experimental results on the synthetic graphs [45]. Once we have generated synthetic graphs, we need to validate if the generated synthetic graph exhibits statistically similar properties to the empirical real-world graphs [32].

The rest of this report is organized as follows: Section 2 briefly introduces the potential applications of synthetic graph generation in graph analysis. Important terms needed to understand this report are described in Section 3. We provide detailed descriptions of three major steps in obtaining realistic synthetic graphs: Section 4 for graph model selection; Section 5 for graph generation; and Section 6 for graph model validation. Section 7 discusses challenges and opportunities across tasks, suggesting future directions. Section 8 concludes the report.

2. APPLICATION DOMAINS AND FUNCTIONAL USES OF SYNTHETIC GRAPHS

We first discuss application domains that employ realistic synthetic graph generation, and go on to the functional (in a more technical sense) uses of the synthetic graph construction steps.

2.1 END-USER DOMAINS

We briefly introduce candidate domains and their rationale for the use of synthetic graphs in order to highlight the necessities of synthetic graphs.

- Healthcare: entities in the healthcare delivery environment are effectively modeled as patient nodes, provider nodes (doctors, hospitals, etc.), and edges for patient-to-provider visits which represent the financial transaction events [13]. When we develop algorithms to identify malicious entities associated with financial fraud, waste, and abuse cases, we have to protect critical information that resides in the data in order to comply with rules like Health Insurance Portability and Accountability Act [42].
- Social Network: social network service providers are likely to impose stricter restriction on the access to their user information to protect the privacy of users, and in turn allow researchers access to only a portion of the actual networks [45]. Still, we would like to develop a valid approach for identifying the most significant user activity patterns in the entire network, though we only have a portion of the actual network.
- Internet Network Topology: it is almost impossible to construct the topology of all the machines connected to the Internet, considering the explosion of the Internet of Things and many virtualized machines in cloud environments [37]. Still, we would like to measure the effectiveness of specific routing algorithms for the next five to ten years.
- Bioinformatics: many problems in bioinformatics, such as Gene Regulatory Networks, can be represented as graphs [15]. These graphs can be very large and intrinsically stochastic. Thus, testing algorithms over representative sampled graphs can be desirable.

2.2 FUNCTIONAL USES

As mentioned earlier, synthetic graphs have two broad functional uses: (1) hiding critical information and (2) controlling the size of graphs. Graph anonymization is one way to prevent the leakage of critical information. As for controlling the size of the graphs, we have three use cases: to generate larger graphs than available empirical graphs (graph extrapolation); to generate smaller graphs than available empirical graphs (graph sampling); and to store a graph with smaller storage requirements (graph compression). We will briefly introduce these applications.

2.2.1 Graph anonymization

Graph anonymization prevents the identity disclosure of individuals in graphs [53]. In graph data, anonymization is a more challenging problem than single-table data since information on individuals can be leaked through observing structural similarities of neighborhoods of nodes [24]. This will be critical for applications like healthcare, national security, and social network anlaysis, where graphs may contain sensitive and private information. Solutions to this problem typically consider specific privacy information

to be protected such as relation identity and individual identity, under the assumption of certain prior information that the adversary owns [53].

Similar to anonymization, differential privacy based approaches exist [26, 46] that apply to graphs. In cryptography, differential privacy aims to provide the means to maximize the accuracy of queries from statistical databases while minimizing the chances of identifying its records. Differential privacy approaches typically consider adding noise to answers on statistical queries over the graph as with the differential privacy on relational data. For instance, Karwa et al. [26] outputs approximated answers to subgraph counting queries to prevent the leakage of privacy from a graph. Differential privacy is a stronger notion to preserve privacy than anonymization, since differential privacy aims to preserve privacy even when additional data is available (i.e. any prior statistical information on the data or a supplemental data related to the target data [41].) Graph anonymization typically modifies the graph data through generalization and perturbation of a graph, while differential privacy typically modifies the answer to certain statistical queries. Graph anonymization is more closely related to the goal of generating calibrated synthetic graphs. More details on graph anonymization can be found in a survey of anonymization techniques on social network data [53].

2.2.2 Graph sampling

Graph sampling typically generates a smaller graph by sampling edges [2] or nodes in a graph [31]. Graph sampling is analogous to sampling non-zero entries of a matrix $\mathbf{A}$ in order to produce a sparse sketch of it, $\mathbf{B}$ to minimize the discrepancy between $\mathbf{A}$ and $\mathbf{B}$ [1]. In addition, graph sampling considers efficiently estimating the graph properties with a known degree of accuracy, by consulting a sample of the whole population [5]. For instance, graph sampling targets at the approximation of other graph properties such as the count of subgraphs like triangles, clustering coefficient, and average degree [3]. Graph sampling is useful for running simulation experiments (simulating routing algorithms in computer networks, or virus/worm propagation algorithms), when these algorithms may be too slow to run on large graphs, [3]. As shown in [32], we can sample a graph, using realistic synthetic graphs with a parameter that controls the size of the graphs, while preserving properties of larger graphs.

2.2.3 Graph compression

In graph compression, a graph compression algorithm A takes the adjacency matrix of a graph as an input and stores it in a compressed data structure [19]. Graph compression is heavily discussed in storing web graphs since storing web graphs involves storing URLs and their hyperlinks. Graph compression often uses the topological structure of a graph such as the similarity of nodes with respect to the set of neighbors for each node [9], which can be used to efficiently encode the node labels [6]. Thus, using graph models, we can compress a graph by capturing the structure of a graph with model parameters and the deviations between the real and the synthetic graph from the generation model [32].

2.2.4 Graph extrapolation

We can use the model to generate larger graphs than we can empirically obtain. Graph extrapolation is useful to understand the evolution of the graph in the future and the characteristics of the graph beyond the obtained empirical graph. For instance, it is prohibitive to construct the communication network topology of the entire Internet [18], and the whole graph of all the users in a social network service [45]. Thus, empirical graphs for those analyses are essentially sampled subgraphs, for which extrapolated graphs are

essential so as to validate that the hypothesis will hold for the entire graph. In order to generate extrapolated graphs, we first need to model the empirical graph and generate a larger graph according to the model, which is exactly the procedure of generating realistic synthetic graphs.

3. TERMINOLOGY

This section highlights some of the key graph theoretic terms pertinent to this survey. Since the generation of synthetic graphs necessarily involves the use of statistical techniques, terminology from the domain of statistical model selection is also included. Note that a distinction is made between the raw (or empirical) data and the synthetic data, which is created or generated via a specific process.

3.1 FUNDAMENTAL GRAPH CONCEPTS

3.1.1 Definitions

At a basic level, a **graph** G is a mathematical structure that can used to represent relationships in data. Let $G = (V, E)$, where $V = 1, 2, \ldots, N$ is the set of vertices (or nodes)[1] and $E \subseteq V \times V$ is the set of edges. While there are many different types of graphs (e.g., multigraphs, directed graphs or weighted graphs), we specifically highlight **attributed graphs**, where attributes are associated with nodes and/or edges. Let α be the node labeling function $\alpha : V \rightarrow L_N$ then G is denoted by $G = (V, E, l)$. If both nodes and edges in a graph have attributes, the attributed graph can be denoted by $G = (V, E, \alpha, \beta)$, where $\alpha : V \rightarrow L_N$, and $\beta : E \rightarrow L_E$ are node and edge labeling functions respectively.

A graph $G^{'} = V^{'}, E^{'}$ is said to be a **subgraph** of G if and only if $V^{'} \subseteq V$ and $E^{'} \subseteq E$. Conversely, if these conditions hold, G is said to be a **supergraph** of $G^{'}$.

Two graphs are said to be **isomorphic** if they are equivalent differing only in their names of vertices and edges. Formally stated, two graphs $G = \{V_1, E_1\}$ and $H = \{V_2, E_2\}$ are isomophic if there is a bijective function (one-to-one correspondence) $f : V_1 \rightarrow V_2$ such that for each edge $\{a, b\} \in E_1$ then there exists the edge $\{f(a), f(b)\} \in E_2$.

3.1.2 Related research and challenges

The problem of testing whether two graphs (or subgraphs) are really the same ((sub)graph isomorphism) is a fundamental graph theory problem known to be NP-hard. When generating synthetic graphs, it is necessary to compare whether the resultant graph is the same as the empirical graph. In so doing, one can measure the **similarity** of two graphs, thereby determining the extent to which they are analogous. This concept comes into play specifically when attempting to fit a model to a set of data. Graph similarity has numerous applications in diverse fields such as social networks, image processing, biological networks, chemical compounds, and computer vision.

In order to perform comparisons between synthetic and raw graphs, some metric is required. A variety of terms can be found throughout the literature including *graph properties, features, metrics* or even *graph statistics*. Broadly speaking, these refer to a quantifiable property of the data that can be used for comparison.

3.2 STATISTICAL MODELING

This section clarifies the terminology related to graph modeling as appropriate for this report.

[1] the terms will be used interchangeably in this report

3.2.1 Definitions

A **graph generator model** is a formalized mathematical approach to creating a graph. While this is closely related to *statistical models*, which represent the data generation process via equations related to random and non-random variables, the graph generation model specifically focuses on forming data which is composed of nodes and edges.

The *graph generator* itself is a specific instance (i.e., code implementation) of a model. For example, the Ërdos-Rényi model has many implementations such as the NetworkX or APGL packages. A graph generator model may be controlled via a set of *parameters*.

Statistical model selection is the process by which, given a set of possible statistical models $\mathbf{M} = \{M_1, M_2, \ldots, M_k\}$, one must pick the model that is expected to most closely replicate the test data, with respect to certain criteria. **Graph generator model selection** is a specific case of this, where $\mathbf{M}$ consists of graph generator models and the target data is a graph.

For parameter controlled models, model selection includes a scan of the parameter space referred to as *parameter search*. With respect to graph models, the term **graph fitting** is also used when referring to the steps involved in selecting an instance of a graph model and the optimal parameters which generate the most similar synthetic graph to the raw data at hand.

In statistics and machine learning, *model validation* or *model evaluation*, is the task of quantifying the quality of prediction from a statistical model with parameters obtained from model selection process. Similarly, **graph generator model validation** is the task of quantifying the quality of synthetic graphs generated.

Finally, the term **fidelity** describes the similarity of the statistical properties of two graphs. It can be both used for graph fitting or graph generation model validation. Fidelity can be defined in many different ways. For instance, let us say that vectors v_{G_1} and v_{G_2} are composed of a set of statistical properties of graphs G_1 and G_2 respectively, then we can use the cosine similarity of these two vectors as *fidelity* of the graphs.

3.2.2 Related research and challenges

The issues of graph fitting and validation each present their own set of challenges. In order to determine if a model provides a "good" match, a mechanism to perform the fitting is required, as are metrics for determining the appropriateness of the selection.

The next section discusses in further detail graph generation models and existing fitting techniques.

4. GRAPH GENERATIVE MODEL SELECTION

Conventionally, *model selection* in statistics or machine learning is the task of finding a good statistical model, including corresponding model parameters, which estimates the real-world phenomenon [54]. In general, a model selection approach takes the following steps. First, it initializes a model by selecting parameters (e.g., through random selection). Then, it searches other candidate parameters according to the parameter search strategy and evaluates the model with the new candidates. By repeatedly performing this parameter search process called *model fitting*, it selects parameters with the best fitness for the observed data. (Curve fitting [28] is an example of a model selection approach.) *Graph generative model selection* can be understood as a special case of model selection. It is the process to search for a good graph generative model, along with parameters that can generate synthetic random graphs with similar statistical properties to the target graph. This section begins with the introduction of popular graph generative

models. After the introduction of graph models, we will cover *graph fitting techniques*, which extract optimal parameters for a graph generative model in order to produce a randomized synthetic graph that matches the statistical properties of a target graph.

4.1 GRAPH GENERATIVE MODELS

We categorize graph generative models into three model groups: *feature-driven models*, *structure-driven models*, and *intent-driven models*.

Early graph generative models fell into the first group which focuses on reproducing one or more specific statistical graph features (e.g., degree distribution, graph density, etc.). Feature-driven models define a mechanism or a principle by which a network with desired features is constructed. Although feature-driven models are simple to use, they are limited in that the selected features and their statistical properties may not fully represent the characteristics of the entire graphs. For instance, two graphs with the same average degree can have very different topological shapes. To address this limitation, structure-driven models were proposed. Unlike the feature-driven models, they focus on capturing the overall structure (or topology) characteristics of empirical graphs, instead of capturing specific global graph properties. These models normally require expensive memory and computational resources and we often need to balance the accuracy of the model with the computational cost associated with running the model. The third category, intent-driven models, includes graph generative models that focus on emulating the process of graph evolution. This approach is based on the assumption that emulating the behaviors in the network (e.g. a user in a social network) will result in realistic graphs resembling observed graphs.

We frame the state of the art in the context of the above three graph generative model categories. (For further discussion on specific graph generative models, readers may want to refer to [36, 39].)

- **Feature-driven Models:**
  - **Erdös-Rényi** [16, 21]: there are two variants of the Erdös-Rényi (ER) model. In the first model [16], a graph is chosen uniformly at random from the set of all possible graphs with given number of nodes and edges. In the second model [21], a graph is generated by connecting nodes randomly with probability p. The model becomes more likely to generate a graph with more edges as the probability p increases. Due to the convenience of simply adding statistically independent edges, the second model is more commonly used in practice. Thus, we can state that ER model can reproduce graphs where the node degree distribution in the generated graph follows Poisson distribution. The ER model has been criticized that it may not be appropriate for modeling many real-world phenomena, since it does not have heavy tails like many real-world graphs.
  - **Watts-Strogatz (small world)** [50]: this model is a random graph generative model that produces graphs with small-world properties – short average path length and high clustering coefficient. The underlying structure of the model is a lattice structure that produces a locally clustered network and random links reduce the average path lengths. The major limitation of this model is that it produces an unrealistic degree distribution in comparison with many real scale-free networks. In addition, since this model typically employs a fixed number of nodes, it cannot be used to model network growth.
  - **Barabasi-Albert** [7]: many real-world graphs are scale-free networks. In other words, they have power-law degree distributions; however, both the Erdös-Rényi model and the Watts-Strogatz model do not generate graphs with power-law degree distribution. The Barabasi-Albert model is proposed to address the limitations of both models. It exploits the incremental growth model for graph construction with the idea that the node attaches

preferentially to nodes that are already well connected. In other words, new objects tend to attach to popular objects. Each new node is connected to existing nodes with a probability that is proportional to the number of links that the existing nodes have. It is also called the *preferential attachment* model. This model's limitation is that it fails to produce the high levels of clustering seen in some real networks.

  - **Krioukov** [29]: this model also generates a graph with power-law degree distribution. The model is based on the idea that there are hidden metric spaces which influence the topology of the network. A distance measure between two nodes can be defined in the hidden space, and the distance affects the probability of connecting the two nodes. (The model uses hyperbolic spaces as the underlying hidden space.)
  - **Forest Fire** [33]: earlier models are bounded by constants; however this model aims to capture the characteristic evolution of real-world graphs over time, particularly focusing on increasing density and shrinking average distance between nodes. When a new node is connected to an existing node, it also randomly connects to some of the node's neighbors.

- **Structure-driven Models:**
  - **dK-Graph** [37]: dK-Graph is based upon dk-Series, which represent reproducible graph properties. dK-Series are probability distributions that specify all degree correlations within d-sized subgraphs of a given graph G. A larger d will capture more complex properties of G. Interpreting dK-series, dK-0 is average node degree, dk-1 captures the node degree distribution, dK-2 captures the joint degree distribution, dk-3 captures the clustering coefficient, and etc. Larger number of d require higher costs for computation. dK-series can be used in two ways. First, it can be exploited to measure the distance between two graphs. Second, it can be used to generate random graphs that accurately reproduce the given metrics. According to the authors, $d = 2$ is sufficient for most practical purposes but this has been debated subsequently. For example, when modeling certain biological networks, $d = 3$ was not deemed a useful metric to capture the realistic characteristics of the graph [49].
  - **Kronecker Graphs** [30]: the objective of the Kronecker graph model is to obey multiple properties of a given original graph. The Kronecker graphs are generated via recursive creation of self-similar graph starting from an initiator adjacency matrix. The evolution process is called Kronecker multiplication and it can be computed in linear time. Each evolution creates successively larger graphs. Sampling and the maximum likelihood approach are used to fit the original graph to the appropriate parameters. The Kronecker model has been selected as one of the models for the Graph 500 benchmark. However, it has several known drawbacks. First, it can be expensive to perform graph model selection when the target real-world graphs contain more than billion edges. The size of the initiator graph and the maximum number of gradient descent iterations in the search are important parameters for this model, but larger parameters can lead to significantly high computational cost. Second, clustering coefficients of generated graphs are much smaller than what is produced in real-world graphs.
  - **Musketeer** [23]: this model aims to produce a randomized synthetic graph as similar to the original graph as possible. The model performs a rewiring process starting from the original graph, trying to preserve intended features of the original graph. There are several transformations such as editing, adding nodes or edges. The process is called *graph coarsening*, and it is repeatedly performed a given number of times by computing projections of the Laplacian. After a series of coarsening steps, the graph structure is refined by randomized edits to finalize the synthetic graph generation.

- **Intent-driven Models:**
  - **Random Walk** [48]: this model emulates the randomized behavior of a user's behavior of friend discovery in social networks. Initially, the graph starts with one vertex with no edges, repeating the following steps. The first step creates a new vertex v_i and connects the vertex v_i to a randomly chosen existing vertex v_j. Next, with a given probability p, the model repeatedly creates another edge connecting v_i to a v_j's direct neighbors until no edge is created. There are recursive versions of the Random Walk model too. A new node is added to the network, then random walk traversals are performed starting from a randomly chosen node in the graph. During the random traversals, the currently visited node is probabilistically attached to the new node.
  - **Nearest Neighbor** [48]: this model emulates the observed behavior in social networks. Two users sharing more common friends are more likely to become friends. When a new node is added to the graph, the node is linked with a randomly chosen existing node, then additional random pairs of nodes that are 2-hops away from the new node are also connected.

4.2 GRAPH FITTING TECHNIQUES

Graph fitting is a task to extract a set of model parameters, which we can feed into a graph generative model to generate randomized synthetic graphs that match the given real-world graph. For this task, we need two components – a *search strategy* of model parameters and a *criterion* to evaluate fitness of models. There are three technical hurdles encountered in this process. First, as indicated earlier, some graph generative models are unnatural to fit real-world graphs. For instance, the Erdős-Rényi model is not able to generate a graph with a power-law distribution; the nearest-neighbor model has been known to always generate graphs with power-law coefficients greater than 2 [45]. In these cases, we may simply not be able to find good parameters for a target graph with the generating model being considered. Second, finding optimal parameters often requires significant computational cost. Graph generative models can have multiple parameters and evaluating a model with a set of parameters can be time-consuming especially when the graphs are large. So, in addition to strategies to find good parameters, we need to carefully consider the balance between accuracy and processing time. Third, often graph fitting techniques are tightly coupled with specific graph generative models so the approaches are only applicable to them. For instance, KronFit [30] is a graph fitting procedure which is specifically designed to find parameters for the Kronecker model. Another example is Musketeer [23] which uses its own fitting algorithm.

We observe that in the context of graphs, model fitting is less established than model selection (for other forms of empirical data) which means that there is no generally *best-known* graph fitting technique for most cases. Nevertheless, if we were to categorize graph fitting approaches at a fairly high level of abstraction, we would divide them into two groups: (1) Maximum Likelihood Estimation (MLE)-based approaches and (2) graph similarity-based approaches.

Maximum Likelihood Estimation (MLE)-based approach: MLE is a widely used approach for fitting a conventional statistical model to observed data. It computes the maximum probability that a model with different parameters generates data exactly matching the observed data. However, direct application of maximum likelihood estimation over a graph is computationally intractable. For a graph G, each node in the vertex set V of G has a unique but randomly assigned index, or node label. Thus, two isomorphic graphs may have different node labels. Thus, when we estimate the likelihood that a given model generated the target graph G, we would have to consider all possible, $|V|!$ permutations of node labels, because the likelihood $P(G)$ is $\sum_\sigma P(G|\sigma)P(\sigma)$ for all permutations σ of $|V|$ nodes. For example, the KronFit algorithm [30] uses MLDE but considers samples of permutations of σ in order to make the problem

computationally tractable. Unfortunately, the computational requirements are still high even when using the sampling technique during likelihood estimation. KronFit takes 2.5 hours with a relatively small real-world graph with $|V| = 75,879$ and $|E| = 508,960$ on a standard desktop [30]. In order to achieve high fidelity, KronFit can take 48 hours for a graph with 3.6 millions of edges on a stand-alone system with 32GB of RAM [45].

Graph similarity-based approach: this approach finds model parameters to maximize the graph similarity between generated synthetic graphs and the target graph. The challenge is to define a quantitative metric to measure graph similarity that can be efficiently computed in a scalable way. Sala et el. [45] present an approach that uses dK-series as a graph similarity metric. We compute the similarity of two graphs by computing the distance between dK vectors of the two graphs. More specifically, the authors use *dk*-2, which captures the joint degree distribution, to achieve tractable performance. In addition, the authors use uniform parameter sampling with adaptive precision to efficiently locate near-optimal parameters in the large parameter space. The authors leave more efficient sampling techniques for future study. The key advantage of this approach is that it is generally applicable to many graph generative models, including Nearest Neighbor [48], Random Walk [48], Forest Fire [33], and Barabasi-Albert [7].

5. GRAPH GENERATION

Generation approaches are often tightly coupled with certain graph generative models. This implies that a graph generative model itself can include the way of generating graphs. For instance, the preferential attachment method is coupled with Barabasi-Albert's model [7]. We discussed the generation approaches for such generative models in Section 4. In this section, we focus on core graph generation techniques that are applicable across multiple graph generative models. For completeness, we also list here available oftware packages that can be utilized for graph generation from an earlier survey.

5.1 GENERATION TECHNIQUES

We categorize generation approaches into four groups: *stochastic*, *pseudograph*, *matching*, and *rewiring*, described as follows:

- **Stochastic**: this approach is the simplest and most convenient for theoretical analysis. In essence, it selects a pair of nodes and connects them with a certain probability, which repeats until all the nodes are considered. Depending on the family of models, the difference centers around the way of calculating the probability of connecting a pair of nodes. For instance, we might generate a graph by connecting every pair of n nodes with probability $p = \bar{d}/n$, where $\bar{d}$ is the average degree and n is the number of nodes in the graph [16]. If we want to utilize degree distribution, we can label all nodes with their expected degrees drawn from the distribution [14]. After labeling all the nodes, we connect a pair of nodes, v_i and v_j, with the probability $p(v_i, v_j) = q(v_i)q(v_j)/\bar{d}$, where $q(v_i)$ and $q(v_j)$ are the expected degrees of node v_i and v_j, respectively. (There exist studies to calculate the probability of connecting nodes according to the joint degree distribution [37].)

  A disadvantage of this approach is that it is not easy to obtain a synthetic graph that matches the target empirical graph with respect to arbitrary graph features. For instance, when we use the degree distribution to calculate the probability of connecting a pair of nodes, a large portion of nodes with expected degree 1 can be connected to nodes with degree 0, resulting in many tiny connected components.

- **Pseudograph (configuration) and matching**: this approach attempts to exactly reproduce given degree distributions, allowing multiple edges between the same pair of nodes and self-looping edges for a node [4]. The approach operates by (1) selecting n_k nodes according to the degree distribution, where k is the degree of a node, (2) attaching k stubs to each of nodes, and (3) randomly choosing pairs of stubs and connect them to form edges. If we desire a graph that does not contain multiple edges between a pair of nodes and self-loops, the method removes offending loops and edges.

  The matching approach differs from the pseudograph approach in avoiding loops during the construction phase. The algorithm works exactly as with the pseudograph approach, but skips pairs of stubs that form loops if they are connected. Mahadevan et al. [37] showed that both approaches can be extended to generate the valid joint degree distribution.

- **Rewiring**: introduced in [50], the main idea of the rewiring approach is to rewire *random* pairs of edges preserving an existing form of distribution such as degree distribution and joint degree distribution, also with cluster coefficient. The rewiring process converges after $O(|E|)$ rewiring steps, producing a graph having desired degree distribution [22]. With rewiring approaches, randomized approximation approach exists to inclusively consider multiple graph properties to generate synthetic graphs [37].

5.2 GENERATOR PACKAGES

Table 1 [36] summarizes the list of available software packages that can be utilized to generate synthetic graphs. From the table, we notice that only a few packages can handle multiple models[2]. Recall that we would like to select a model among numerous candidate graph generative models. Therefore, we will need to employ numerous individual packages to select a single model, which will cause inconvenience over the procedure as each package might favor different dependent libraries and system architectures. Additionally, most of these packages are not designed for large scale parallel or distributed systems, with a few exceptions like the generators in Graph500. For mode details on generators (both packages and generation algorithms), we refer readers to [12, 36].

6. GRAPH GENERATIVE MODEL VALIDATION

Validating a graph model is the step that checks if a generated synthetic graph matches the target empirical graph with respect to graph features of interest. We might have already employed the graph properties of interest as a fitting criterion during the graph model selection step. However, we typically consider more graph features in the validation task than in the model selection step. It is primarily because of the different natures of these two processes – model selection involves iterative computation, graph fitting, and searching the parameter space, while graph generative model validation can be a process without iterations. For an iterative process like fitting, considering multiple graph properties is computationally intensive [34]. Therefore, systems consider fewer graph properties during the fitting process but validate the fitted graph against more graph features during the graph model validation step. We list a few popular and desired features which could be used to evaluate the fidelity or similarity between the generated synthetic graphs and the target graph.

- Spectrum: the set of eigenvalues of the adjacency matrix. (In physics, the eigenvalues of the Laplacian matrix of a graph are sometimes called the graph's spectrum.)

[2]Tests were run on a 48-core HP DL585 G7 with 384GB of RAM.

Table 1. List of generator packages and supported graph models [36].

Package	Supported models	Limit	Limiting factor
APGL	Configuration model	37 million vertices	
	Erdos-Renyi	2^{18} vertices	time
	Prefential Attachment	2^{16}	memory/time
	Watts-Strogatz	2^{16}	time
Boost	PLRG	2^{30} vertices	memory
BTER	Erdos-Renyi (block 2-level)	2^{28} vertices	memory
ergm	ERGM	$\sqrt{\frac{2^{32}-1}{2}}$ vertices	max integer ($\frac{2^{32}-1}{2}$)
GGEN	Erdos-Renyi (gnp method/layer method)	2^{18} vertices	memory
Graph500	Kronecker (R-Mat+noise)	2^{30} vertices	memory
gt-item	Tiers	2^{26} vertices	memory
INDDGO	Partial k-tree	2^{37} vertices	memory
inet-3.0	Inet	$3037 < n \leq 2^{17}$	time/model limitation
Krioukov	Hyperbolic	2^{20} vertices	time
MFR	RDPG	< 46000 vertices	memory
Musketeer	Musketeer	2^{18} vertices with 7% and 8% edit rates	time
NetworkX	Erdos-Renyi (original/fast variant)	2^{22} vertices	time
	Preferential Attachment	2^{23} vertices	time
	Watts-Strogatz	2^{24} vertices	time
	Waxman	8000 vertices with default $a = 0.4$, $b = 0.1$	time
pywebgraph	Kronecker (R-Mat)	2^{20} vertices	time
stocksim	Waxman	< 180000 vertices	memory
tier	Tiers	2^{26} vertices	memory

- Distance Distribution $d(x)$: the number of pairs of nodes at a distance x, divided by the total number of pairs. It is used for evaluating the performance of routing algorithms or measuring speed of worms spread in a network.
- Betweenness Centrality: a weighted sum of the number of shortest paths passing through a given node or link. It is used to estimate traffic on a node or link.
- Likelihood S: the sum of products of degrees of adjacent nodes. It is linearly related to the assortativity coefficient which is a measure of how similar or dissimilar nodes tend to interconnect.
- Degree Distribution: the probability of nodes having degree k in a graph. Many real-world graphs have power law degree distributions.
- Clustering $C(k)$: a measure of how close neighbors of the average k-degree node are to forming a clique.

In order to measure fidelity or to validate a synthetic graph, we need to compute a set of graph properties and a metric to measure the deviation of two graphs according to the set of graph properties. To measure the fidelity, authors in [45] proposed the Euclidean distance of two feature vectors, where each element corresponds to node degree distribution, joint degree distribution, clustering coefficient, average path length, network radius, and network diameter. They also propose using graph properties as an element of the feature vector of the graph and calculate the $L2$ norm as a fidelity metric between the synthetic graph and the target graph.

7. DISCUSSION: CHALLENGES AND OPPORTUNITIES

In the context of the surveyed material, this section summarizes the outstanding challenges and opportunities in creating realistic synthetic graphs from empirical graphs.

7.1 DEVISING FITTING MECHANISMS AND CRITERIA

In graph generative model selection, the biggest challenge remains devising a promising graph fitting mechanism. Existing graph generative models have overwhelmingly aimed to capture either specific graph features or the way of evolving general real-world graphs such as the Internet, the web, or social network. There has been little to no focus on creating synthetic graphs based on real empirical graphs. The search for model parameters is either rudimentary to be applicable to a specific graph generation approach, or not applicable at all for most existing generative models. We believe the two important gaps to fill are: (1) developing graph fitting mechanisms for each existing graph generative model, and (2) developing new graph generative modeling approaches suitable for fitting synthetic graphs to given empirical graphs.

A key area of future work is to establish computationally-feasible fitting criteria specialized for graph model selection. Establishing fitting criteria remains a promising avenue for future developments. A few potential approaches toward establishing fitting criteria for the graph models are: (1) exploiting dK-series further, and (2) implementing scalable fitting algorithms by taking advantages of recent parallel computing software or hardware platforms. For instance, generating a Kronecker graph involves the Kronecker product (also known as a tensor product) whose scalable algorithm has been well studied. We intend to take advantage of parallel state-of-the-art systems such as Cray's Urika systems deployed at ORNL to run such implementations in a scalable manner.

7.2 DEFINING COMPREHENSIVE FIDELITY

The lack of comprehensive fidelity measures is another major gap in the graph model validation process. In validating graph models, we compare two graphs with respect to multiple graph metrics. Considering each graph metric as an element of a feature vector for a graph, we can consider similarity metrics between two vectors. However, some open questions remain: which graph properties should be selected for the feature vector, and how to measure the similarity between vectors? This topic has not been explored in depth and will benefit from exploiting existing graph similarity measures or developing new measures and developing a similarity comparison technique in the context of graph model validation.

A few possible ways to devise criteria to measure fidelity include:

- Similarity Measures: this will be the method to compare two feature vectors from two graphs as with the use of $L2$ norm in [45]. As another candidate for measuring similarity, we may consider cosine similarity and dot product measures.

- Precision/Recall: similar to information retrieval or pattern recognition, we might use the probability of retrieving the relevant items (precision) and the probability of retrieving the items that exists in the target data (recall). We can extend this concept for the context of graph model validation, using the node correspondence or node labeling problem. (The node correspondence problem is to find the best mapping between nodes of two graphs.)

- Cross Validation: we might compare graph properties of realistic synthetic graphs with other samples of empirical graphs in the same domain (or same population) than the target empirical graphs used in the graph model selection step.

7.3 ACHIEVING SCALABILITY

Scalability is a common challenge in the three main steps taken to obtain realistic synthetic graphs. Recall that scalability has been one of the key concerns in computing graph properties from a graph [25, 38, 44, 51] for years. In graph model selection, we employ graph properties for fitting criteria (degree distribution, counts of subgraphs, etc.) and, in graph model validation, we employ graph properties for elements of fidelity measures. Therefore, it is natural that scalability becomes a major concern in both steps.

For graph model selection, existing work employs a limited number of graph features - typically less than four. The demand of scalability stems from the fact that fitting is an iterative process. For some graph models, the fitting process may involve sampling subgraphs as with KronFit [32], which can be performed as many times as the number of edges in the graph, where sampling subgraphs is another computational challenge. A gab is the absence of parallel algorithms for either the fitting process or the sampling process.

Furthermore, we have observed that the upper bound on the number of vertices of a graph is typically a billion nodes for many graph generation packages to generate graphs. This is mainly because those packages are designed for a stand-alone single-node system. Considering that a driving demand of realistic synthetic graphs is the need for extrapolating a larger graph than the available empirical graphs, the field would benefit from a scalable parallelized graph generator.

7.4 APPLICATION-LEVEL BENCHMARKS

Many techniques target general graphs or graphs that are not related to any specific application domains. For example, we may not be sure that an e-mail exchange network can be synthetically generated

when we used a graph model selection procedure that worked for the Internet topology. The best fitted model can be different depending on the characteristics of the data and application algorithms. When we measure application-level fidelity, we perform the same benchmark over both the target graph and the synthetically generated graphs. Then, we compare the results to quantify each graph's fidelity. Compared with known graph metrics, these application-level benchmarks can effectively capture the set of "important features" in each domain problem of social networks, which might not be correlated with a single graph property. Some examples of such application-level benchmarks are RE [20], Sybilguard [52], and Social Shield Anonymous System [43].

7.5 MODEL SELECTION FOR HETEROGENEOUS (ATTRIBUTED) GRAPHS

Most existing graph models and graph generators assume homogeneous graphs which are composed of single types of nodes or edges. However, as heterogeneous graphs composed of multiple node and edge types grow, and applications that use these graphs proliferate, the application-aware approach to generating synthetic heterogeneous graphs will have to mature. This topic is a fertile area for new method development.

8. CONCLUSIONS

This report surveyed methods to generate realistic synthetic graphs. The goal of generating realistic synthetic graphs is to create graphs that allow discovering the same knowledge as in the empirical graphs with reasonable statistical confidence with respect to algorithmic operations and infrastructure operations. This report organized the overall approach to generate realistic synthetic graphs into three steps: graph model selection, graph generation, and graph model validation. We found that scalability is a common issue across all three steps, while each step has additional unique challenges. The graph model selection step needs efficient parameter search strategies and fitting criteria to evaluate the fitness of models; and the graph validation step requires accurate fidelity measurement methods, which are currently open research problems.

ACKNOWLEDGMENTS

This work was supported by the United States Department of Defense (DoD) and used resources of the DoD-HPC Program at Oak Ridge National Laboratory.

References

[1] Dimitris Achlioptas, Zohar S. Karnin, and Edo Liberty. Near-optimal entrywise sampling for data matrices. In C.j.c. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 1565–1573. 2013.

[2] Nesreen K. Ahmed, Nick Duffield, Jennifer Neville, and Ramana Kompella. Graph sample and hold: A framework for big-graph analytics. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 1446–1455, New York, NY, USA, 2014. ACM.

[3] Nesreen K. Ahmed, Jennifer Neville, and Ramana Kompella. Network sampling: From static to streaming graphs. *ACM Trans. Knowl. Discov. Data*, 8(2):7:1–7:56, June 2013.

[4] William Aiello, Fan Chung, and Linyuan Lu. A random graph model for massive graphs. In *Proceedings of the Thirty-second Annual ACM Symposium on Theory of Computing*, STOC '00, pages 171–180, New York, NY, USA, 2000. ACM.

[5] Edoardo M. Airoldi, Xue Bai, and Kathleen M. Carley. Network sampling and classification: An investigation of network model representations. *Decis. Support Syst.*, 51(3):506–518, June 2011.

[6] Alberto Apostolico and Guido Drovandi. Graph compression by bfs. *Algorithms*, 2(3):1031–1044, 2009.

[7] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *science*, 286(5439):509–512, 1999.

[8] Albert-Laszlo Barabasi and Zoltan N Oltvai. Network biology: understanding the cell's functional organization. *Nature Reviews Genetics*, 5(2):101–113, 2004.

[9] Paolo Boldi, Marco Rosa, Massimo Santini, and Sebastiano Vigna. Layered label propagation: A multiresolution coordinate-free ordering for compressing social networks. In *Proceedings of the 20th International Conference on World Wide Web*, WWW '11, pages 587–596, New York, NY, USA, 2011. ACM.

[10] Vincenzo Bonnici, Rosalba Giugno, Alfredo Pulvirenti, Dennis Shasha, and Alfredo Ferro. A subgraph isomorphism algorithm and its application to biochemical data. *BMC Bioinformatics*, 14(S7), 2013.

[11] Nathan Brown. Chemoinformatics: an introduction for computer scientists. *ACM Comput. Surv.*, 41(2):8:1–8:38, February 2009.

[12] Deepayan Chakrabarti and Christos Faloutsos. Graph mining: Laws, generators, and algorithms. *ACM Computing Surveys*, 38(1):2, 2006.

[13] Varun Chandola, Sreenivas R. Sukumar, and Jack C. Schryver. Knowledge discovery from massive healthcare claims data. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '13, pages 1312–1320, New York, NY, USA, 2013. ACM.

[14] Fan Chung and Linyuan Lu. Connected components in random graphs with given expected degree sequences. *Annals of combinatorics*, 6(2):125–145, 2002.

[15] Hidde De Jong. Modeling and simulation of genetic regulatory systems: a literature review. *Journal of computational biology*, 9(1):67–103, 2002.

[16] Paul Erdös and Alfréd Rényl. On random graphs I. *Publ. Math. Debrecen*, 6:290–297, 1959.

[17] Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos. On power-law relationships of the internet topology. In *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, SIGCOMM '99, pages 251–262, New York, NY, USA, 1999. ACM.

[18] Sally Floyd and Eddie Kohler. Internet research needs better models. *ACM SIGCOMM Computer Communication Review*, 33(1):29–34, 2003.

[19] Hana Galperin and Avi Wigderson. Succinct representations of graphs. *Information and Control*, 56(3):183–198, 1983.

[20] Scott Garriss, Michael Kaminsky, Michael J. Freedman, Brad Karp, David Mazières, and Haifeng Yu. Re: Reliable email. In *Proceedings of the 3rd Conference on Networked Systems Design & Implementation - Volume 3*, NSDI'06, pages 22–22, Berkeley, CA, USA, 2006. USENIX Association.

[21] Edgar N Gilbert. Random graphs. *The Annals of Mathematical Statistics*, pages 1141–1144, 1959.

[22] Christos Gkantsidist, Milena Mihail, and Ellen Zegura. The markov chain simulation method for generating connected power law random graphs. In *Proceedings of the Fifth Workshop on Algorithm Engineering and Experiments*, volume 111, page 16. SIAM, 2003.

[23] Alexander Gutfraind, Lauren Ancel Meyers, and Ilya Safro. Multiscale network generation. *arXiv preprint arXiv:1207.4266*, 2012.

[24] Michael Hay, Gerome Miklau, David Jensen, Philipp Weis, and Siddharth Srivastava. Anonymizing social networks. *Computer Science Department Faculty Publication Series*, page 180, 2007.

[25] U Kang, Charalampos E Tsourakakis, and Christos Faloutsos. Pegasus: A peta-scale graph mining system implementation and observations. In *Data Mining, 2009. ICDM'09. Ninth IEEE International Conference on*, pages 229–238. IEEE, 2009.

[26] Vishesh Karwa, Sofya Raskhodnikova, Adam Smith, and Grigory Yaroslavtsev. Private analysis of graph structure. *ACM Trans. Database Syst.*, 39(3):22:1–22:33, October 2014.

[27] Jon M. Kleinberg, Ravi Kumar, Prabhakar Raghavan, Sridhar Rajagopalan, and Andrew S. Tomkins. The web as a graph: Measurements, models, and methods. In *Proceedings of the 5th Annual International Conference on Computing and Combinatorics*, COCOON'99, pages 1–17, Berlin, Heidelberg, 1999. Springer-Verlag.

[28] William M Kolb. *Curve fitting for programmable calculators*. Imtec, 1984.

[29] Dmitri Krioukov, Fragkiskos Papadopoulos, Maksim Kitsak, Amin Vahdat, and Marián Boguná. Hyperbolic geometry of complex networks. *Physical Review E*, 82(3):036106, 2010.

[30] Jure Leskovec, Deepayan Chakrabarti, Jon Kleinberg, Christos Faloutsos, and Zoubin Ghahramani. Kronecker graphs: An approach to modeling networks. *J. Mach. Learn. Res.*, 11:985–1042, March 2010.

[31] Jure Leskovec and Christos Faloutsos. Sampling from large graphs. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, pages 631–636, New York, NY, USA, 2006. ACM.

[32] Jure Leskovec and Christos Faloutsos. Scalable modeling of real graphs using kronecker multiplication. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pages 497–504, New York, NY, USA, 2007. ACM.

[33] Jure Leskovec, Jon Kleinberg, and Christos 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*, KDD '05, pages 177–187, New York, NY, USA, 2005. ACM.

[34] Seung-Hwan Lim, Sangkeun Lee, G. Ganesh, Tyler C. Brown, and Sreenivas R. Sukumar. Graph processing platforms at scale: Practices and experiences. In *Proceedings of the 2015 IEEE International Symposium on Performance Analysis of Systems and Software*, 2015.

[35] H Linhart and W Zucchini. *Model Selection*. John Wiley & Sons, Inc., New York, NY, USA, 1986.

[36] Joshua Lothian, Sarah Powers, Blair D. Sullivan, Matthew Baker, Jonathan Schrock, and Stephen W. Poole. Synthetic graph generation for data-intensive hpc benchmarking: Background and framework. Technical Report ORNL/TM-2013/339, Oak Ridge National Laboratory, 2013.

[37] Priya Mahadevan, Dmitri Krioukov, Kevin Fall, and Amin Vahdat. Systematic topology analysis and generation using degree correlations. In *ACM SIGCOMM Computer Communication Review*, volume 36, pages 135–146. ACM, 2006.

[38] Grzegorz Malewicz, Matthew H Austern, Aart JC Bik, James C Dehnert, Ilan Horn, Naty Leiser, and Grzegorz Czajkowski. Pregel: a system for large-scale graph processing. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, pages 135–146. ACM, 2010.

[39] Michael Mitzenmacher. A brief history of generative models for power law and lognormal distributions. *Internet mathematics*, 1(2):226–251, 2004.

[40] Mark EJ Newman, Duncan J Watts, and Steven H Strogatz. Random graph models of social networks. *Proceedings of the National Academy of Sciences*, 99(suppl 1):2566–2572, 2002.

[41] Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. Smooth sensitivity and sampling in private data analysis. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 75–84. ACM, 2007.

[42] U.S. Department of Health and Human Services. `http://www.hhs.gov/ocr/privacy/`.

[43] Krishina Puttaswama, Alessandra Sala, and Ben Zhao. Improving anonymity using social links. In *IEEE International Workshop on Secure Network Protocols*, 2008.

[44] Lu Qin, Jeffrey Xu Yu, Lijun Chang, Hong Cheng, Chengqi Zhang, and Xuemin Lin. Scalable big graph processing in mapreduce. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, 2014.

[45] Alessandra Sala, Lili Cao, Christo Wilson, Robert Zablit, Haitao Zheng, and Ben Y. Zhao. Measurement-calibrated graph models for social network experiments. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, pages 861–870, New York, NY, USA, 2010. ACM.

[46] Alessandra Sala, Xiaohan Zhao, Christo Wilson, Haitao Zheng, and Ben Y Zhao. Sharing graphs using differentially private graph models. In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, pages 81–98. ACM, 2011.

[47] Peter Uetz, Loic Giot, Gerard Cagney, Traci A Mansfield, Richard S Judson, James R Knight, Daniel Lockshon, Vaibhav Narayan, Maithreyan Srinivasan, Pascale Pochart, et al. A comprehensive analysis of protein–protein interactions in saccharomyces cerevisiae. *Nature*, 403(6770):623–627, 2000.

[48] Alexei Vázquez. Growing network with local rules: Preferential attachment, clustering hierarchy, and degree correlations. *Physical Review E*, 67(5):056104, 2003.

[49] Wenhui Wang, Juan Nunez-Iglesias, Yihui Luan, and Fengzhu Sun. Usefulness and limitations of dk random graph models to predict interactions and functional homogeneity in biological networks under a pseudo-likelihood parameter estimation approach. *BMC bioinformatics*, 10(1):277, 2009.

[50] Duncan J Watts and Steven H Strogatz. Collective dynamics of small-world networks. *nature*, 393(6684):440–442, 1998.

[51] Reynold S. Xin, Joseph E. Gonzalez, Michael J. Franklin, and Ion Stoica. Graphx: a resilient distributed graph system on spark. In *First International Workshop on Graph Data Management Experiences and Systems, GRADES 2013, co-loated with SIGMOD/PODS 2013, New York, NY, USA, June 24, 2013*, page 2, 2013.

[52] Haifeng Yu, Michael Kaminsky, Phillip B. Gibbons, and Abraham Flaxman. Sybilguard: Defending against sybil attacks via social networks. In *Proceedings of the 2006 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM '06, pages 267–278, New York, NY, USA, 2006. ACM.

[53] Bin Zhou, Jian Pei, and WoShun Luk. A brief survey on anonymization techniques for privacy preserving publishing of social network data. *ACM SIGKDD Explorations Newsletter*, 10(2):12–22, 2008.

[54] Walter Zucchini. An introduction to model selection. *Journal of mathematical psychology*, 44(1):41–61, 2000.