Algorithms **2015**, *8*, 1143-1174; doi:10.3390/a8041143

OPEN ACCESS

algorithms

ISSN 1999-4893

www.mdpi.com/journal/algorithms

Article

Generating Realistic Labelled, Weighted Random Graphs †

Michael Charles Davis [1,*], Zhanyu Ma [2,*], Weiru Liu [3], Paul Miller [3], Ruth Hunter [4] and Frank Kee [4]

[1] Organisation Européene pour la Recherche Nucléaire (CERN), Route de Meyrin 385, 1217 Meyrin, Switzerland

[2] Pattern Recognition and Intelligent Systems (PRIS) Lab, Beijing University of Posts and Telecommunications (BUPT), 100876 Beijing, China

[3] School of Electrical and Electronic Engineering and Computer Science, Queen's University Belfast, University Road, Belfast BT7 1NN, UK; E-Mails: w.liu@qub.ac.uk (W.L.); p.miller@qub.ac.uk (P.M.)

[4] Centre for Public Health, Queen's University Belfast, University Road, Belfast BT7 1NN, UK; E-Mails: ruth.hunter@qub.ac.uk (R.H.); f.kee@qub.ac.uk (F.K.)

† This paper is an extended version of our paper published in New Frontiers in Mining Complex Patterns—Second International Workshop (NFMCP 2013), held in conjunction with the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD 2013), Prague, Czech Republic, 27 September 2013.

* Authors to whom correspondence should be addressed; E-Mails: michael.davis@cern.ch (M.C.D.); mazhanyu@bupt.edu.cn (Z.M.); Tel.: +41-75-411-7224 (M.C.D.); +86-134-6632-3341 (Z.M.).

Academic Editor: Maurizio Patrignani

Received: 1 June 2015 / Accepted: 20 November 2015 / Published: 8 December 2015

Abstract: Generative algorithms for random graphs have yielded insights into the structure and evolution of real-world networks. Most networks exhibit a well-known set of properties, such as heavy-tailed degree distributions, clustering and community formation. Usually, random graph models consider only structural information, but many real-world networks also have labelled vertices and weighted edges. In this paper, we present a generative model for random graphs with discrete vertex labels and numeric edge weights. The weights are represented as a set of Beta Mixture Models (BMMs) with an arbitrary number of mixtures, which are learned from real-world networks. We propose a Bayesian Variational Inference (VI) approach, which yields an accurate estimation while keeping computation times tractable. We compare our approach to state-of-the-art random labelled

graph generators and an earlier approach based on Gaussian Mixture Models (GMMs). Our results allow us to draw conclusions about the contribution of vertex labels and edge weights to graph structure.

Keywords: network models; generative algorithms; random graphs; labelled graphs; weighted graphs; bayesian estimation; maximum likelihood estimation; beta distribution; mixture modeling; variational inference

1. Introduction

Network analysis is concerned with finding patterns and anomalies in real-world graphs, such as social networks, computer and communication networks, or biological and ecological processes. Real graphs exhibit a number of interesting structural and evolutionary properties, such as the formation of a giant component, heavy-tailed degree distribution, small diameter, shrinking diameter, and the Densification Power Law (DPL) [1–5].

Besides discovering network properties, researchers are interested in the mechanisms of network formation. Generative graph models provide an abstraction of how graphs form: if the model is accurate, generated graphs will obey the same properties as real graphs. Generated graphs are also useful for simulation experiments, hypothesis testing and making predictions about graph evolution or missing graph elements. Most generative models are for unlabelled, unweighted graphs [1,3,4,6,7], although a few models take discrete vertex labels into account [8–10].

In this paper, we develop a generative model for labelled, weighted graphs. Weights are commonly used to represent the number of occurrences of each edge: e-mails sent between individuals in a social network [11]; calls to a subroutine in a software call graph [12]; or people walking between a pair of sensors in a building access control network [13]. In other applications, the edge weight may represent continuous values: donation amounts in a bipartite graph of donors and political candidates [11]; distance or speed in a transportation network [12]; or elapsed time to walk between the sensors in the building network [13]. In some cases, the weight has more than one dimension [12,13].

Our main motivation for this work is to create "realistic" random graphs for evaluating graph mining algorithms. Some interesting graph datasets are very small; our approach can generate large graphs with the same characteristics as a smaller input graph. Random graphs can also ameliorate concerns about privacy. A second motivation is to better understand the laws governing the relationship between graph structure and attributes. Our experiments show the extent to which various graph properties are related to labels and weights.

Our model, AGWAN (Attribute Graph: Weighted and Numeric), represents the distribution of edge weights as Beta Mixture Models (BMMs) which are learned from weighted input graphs. Learning BMM parameters using Bayesian estimation is analytically intractable. Numerical solutions to simulate the posterior distribution are available, but these incur high computational cost. In Section 3, we introduce an approximation to the prior/posterior distribution of the parameters in the beta distribution and propose an analytically tractable (closed-form) Bayesian approach to parameter estimation, based

on the Variational Inference (VI) framework. Following the principles of VI and utilizing the relative convexity bound, the extended factorised approximation method is applied to approximate the distribution of the BMM parameters. In a fully Bayesian model where all the parameters of the BMM are considered as variables and assigned proper distributions, our approach can asymptotically find the optimal estimate of the parameters of the posterior distribution. In addition, the model complexity depends on the empirical distribution of the data. A closed-form solution is proposed to avoid the need for numerical methods in each iteration. Our approach avoids the drawback of overfitting, as in the conventional Expectation Maximisation (EM) algorithm.

This paper is arranged as follows: Section 2 is an overview of generative graph models and approaches to estimating mixture model parameters; Section 3 presents AGWAN, our generative model for weighted and numeric labelled graphs, including our algorithm for fitting AGWAN's parameters to real input graphs. Section 4 gives an overview of the datasets used in the experiments, and outlines the statistical measures and tests used to evaluate the generated graphs. The experiments in Section 5 demonstrate that the vertex labels and edge weights of a graph can predict the graph structure with high accuracy. Conclusions are in Section 6.

2. Related Work

Our understanding of the mathematical properties of graph structure was pioneered by Paul Erdős and Alfréd Rényi [3]. Graph formation is modelled as a Bernoulli process, parameterised by the number of vertices and a wiring probability between each vertex pair. While it has been essential to our understanding of component sizes and expected diameter, the Erdős-Rényi model does not explain other important properties of real-world graphs such as degree distribution, transitivity and clustering [2,5].

Barabási and Albert's Preferential Attachment model [1] uses the "rich get richer" principle to grow graphs from a few vertices up to the desired size. The probability of an edge is proportional to the number of edges already connected to a vertex. This generates graphs with power-law degree distributions. A number of variants of Preferential Attachment have been proposed [2,5]. Still, Preferential Attachment models lack some desired properties, such as community structure.

The RMat algorithm [6] solves the community structure problem with its recursive matrix approach. RMat graphs consist of 2^n vertices and E edges, with four probabilities a, b, c, d to determine in which quadrant of the adjacency matrix each edge falls. These parameters allow the specification of power-law or log-normal degree distributions; if $a = b = c = d$, the result will be an Erdős-Rényi graph.

Kronecker Graphs [7] fulfil all the properties mentioned above, as well as the DPL (Densification Power Law) and shrinking diameter effect. The model starts with an initiator matrix. Kronecker multipication is recursively applied to yield the final adjacency matrix of the desired size. This work synthesises the previous work in random graphs in a very elegant way and proves that RMat graphs are a special case of Stochastic Kronecker graphs.

The models above tend to have a small number of parameters and are analytically tractable, with simple and elegant proofs of the desired properties. However, graph labels are not taken into consideration. Stochastic models are another class of generative algorithm which may not be amenable to analytical proofs but can be fit to real-world labelled graphs and used to learn the properties of

those graphs. Models in this category include the Stochastic Block Model [10] and Latent Space approaches [8].

The Multiplicative Attribute Graph (MAG) model [9] draws on both of the above strands of research. MAG is parameterised by the number of vertices, a set of prior probabilities for vertex label values and a set of affinity matrices specifying the probability of an edge conditioned on the vertex labels. The affinity matrices can be learned from real graphs using Maximum Likelihood Estimation [14]. [9] proves that Kronecker Graphs are a special case of MAG graphs, and that suitably-parameterised MAG graphs fulfil all the desired properties: log-normal or power-law degree distribution, small diameter, the existence of a unique giant component and the DPL. However, the MAG model can only generate unweighted graphs. We believe that our method, described in the next section, is the first generative model for labelled, weighted graphs.

The AGWAN model requires a suitable probability distribution to model the edge weights accurately and efficiently. The Gaussian distribution is popular as it has an analytically tractable Probability Density Function (PDF); a weighted mixture of Gaussian components provides a reasonable approximation to any general probability distribution [15]. The resulting Gaussian Mixture Model (GMM) is quite flexible and is used extensively in statistical pattern recognition [16]. If the number of mixture components is known in advance, the GMM parameters can be estimated using EM (Expectation Maximisation) [17]. However, if the number of mixtures is unknown, EM can result in overfitting. The problem of knowing the "correct" number of components can be avoided by using a non-parametric model: the approach in [18] assumes an infinite number of components and uses VI (Variational Inference) to determine the optimal number for a given dataset. The variational algorithm can be accelerated for higher-dimensional data using a kd-tree [19].

In [20], the edge weight parameters are specified as a GMM. However, the Gaussian distribution may not be the best choice where the weight is a countable quantity representing the number of occurrences of an edge. In this case, the weights are bounded by $(0, +\infty)$, while the Gaussian distribution is bounded by $(-\infty, +\infty)$. Although the weights can be modelled as a GMM, a large number of mixture components are required to describe the data close to the boundary [21]. Alternatives to the GMM include the truncated GMM [21] and the BMM [22]. We investigate these options in this paper.

The most central task in modeling the data with a BMM is parameter estimation. Since the normalisation constant (the beta function) in the beta distribution is defined as a fraction of integrals, it is difficult to obtain a closed-form expression for estimating the parameters. For maximum likelihood estimation of the BMM parameters, [23,24] proposed an EM algorithm [25], with iterative numerical calculations in the maximisation step. As with GMMs, the EM algorithm for BMM has some disadvantages: it can lead to overfitting when the mixture models are excessively complex; and the iterative numerical calculation in the maximisation step (e.g., with the Newton-Raphson method) has a high computational cost.

For Bayesian estimation, we can formally find the prior distribution and the conjugate posterior distribution of the parameters of the beta distribution. However, this posterior distribution is still defined with an integration expression in the denominator such that the closed-form of the posterior distribution is analytically intractable. [22] proposed a practical Bayesian estimation algorithm based on the Gibbs sampling method, which simulates the posterior distribution approximately rather than computing it.

This method prevents the overfitting problem but still suffers from high computational cost because of the Gibbs sampling, especially when the data is in a high-dimensional space. To overcome this problem, [26,27] proposed a full Bayesian estimation method for parameter estimation in a BMM, where the VI framework was employed and an analytically tractable solution can be obtained. The proposed method facilitates the parameter estimation.

3. AGWAN: A Generative Model for Labelled, Weighted Graphs

In this section, we present our generative model, AGWAN (Attribute Graph: Weighted and Numeric). The model is illustrated in Figure 1.

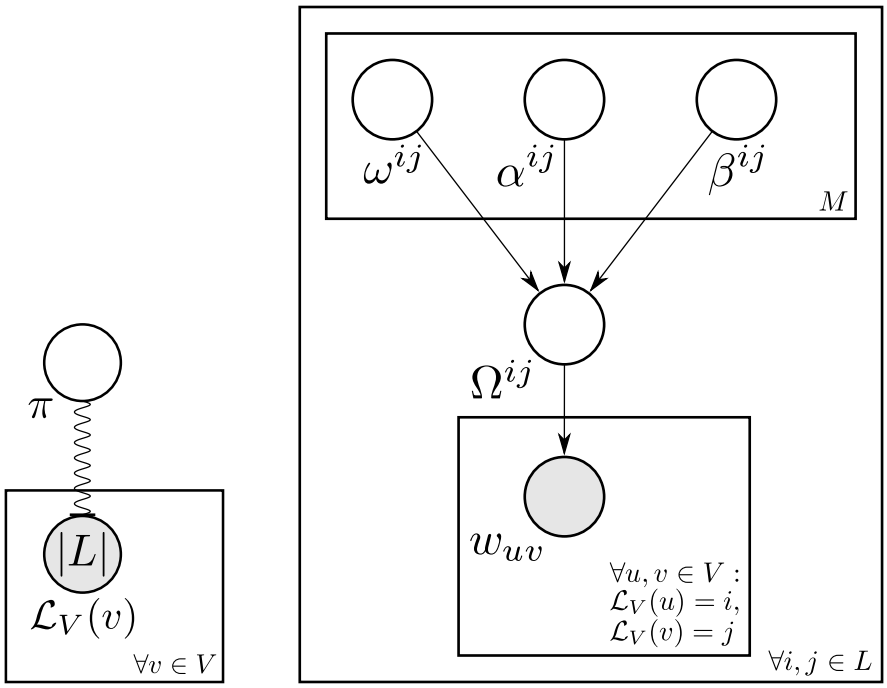


Figure 1. AGWAN parameters. (**a**) Vertex Labels; (**b**) Edge Weights.

Consider a graph $G = (V, E)$ with discrete vertex label values drawn from a set L (Figure 1a). For each combination of vertex attributes $\langle i, j \rangle$, the corresponding mixture model Ω^{ij} parameterises the distribution of edge weights, with an edge weight of 0 indicating no edge (Figure 1b). $u, v \in V$ are vertices and $w_{uv}, w_{vu} \in \mathbb{N}$ are edge weights. Edges $e \in E$ are specified as a 3-tuple $\langle u, v, w_{uv} \rangle$. Thus, the AGWAN model is parameterised by π, the set of prior probabilities over L; and the set of edge weight mixture parameters $\Theta = \{\Omega^{ij} | i, j \in L\}$. For directed graphs, $|\Theta| = |L|^2$ and we need to generate both w_{uv} and w_{vu}. For undirected graphs, $\Omega^{ij} = \Omega^{ji}$, so $|\Theta| = O(|L|^2/2)$ and $w_{vu} = w_{uv}$.

Ω^{ij} is specified as a BMM:

$$\Omega^{ij} = \sum_{m=1}^{M} \omega_m^{ij} \cdot \text{Beta}(a_m^{ij}, b_m^{ij}) \tag{1}$$

where ω_m^{ij} is the weight of each component; and $\mathrm{Beta}(a_m^{ij}, b_m^{ij})$ is the beta distribution with shape parameters (a_m^{ij}, b_m^{ij}). The PDF of the beta distribution is:

$$\mathrm{Beta}(x; a, b) = \frac{1}{\mathrm{beta}(a, b)} x^{a-1}(1-x)^{b-1},\ a, b > 0 \tag{2}$$

where $\mathrm{beta}(a, b)$ is the beta function $\mathrm{beta}(a, b) = \frac{\Gamma(a)\Gamma(b)}{\Gamma(a+b)}$ and $\Gamma(\cdot)$ is the gamma function defined as $\Gamma(z) = \int_0^\infty t^{z-1}e^{-t}dt$. As the support of the BMM is $(0, 1)$, we use a constant s^{ij} to scale the data into this range before fitting the BMM parameters. During graph generation, we draw values from the BMM and multiply by s^{ij}.

We specify Ω^{ij} such that the weight of the first component ($m = 0$) encodes the probability of no edge: $\omega_0^{ij} = 1 - P(e_{ij})$, where $P(e_{ij})$ is the probability of an edge between pairs of vertices with labels $\langle i, j\rangle$ and is learned from the input graph. The weights of the BMM components ($m \in [1, M]$) are normalised by $P(e_{ij})$, so the weights of all the components form a probability distribution: $\sum_{m=0}^{M} \omega_m^{ij} = 1$.

3.1. Graph Generation

Algorithm 1 describes how to generate a random graph using AGWAN(N, L, π, Θ). The number of vertices in the generated graph is specified by N. After assigning discrete label values to each vertex (lines 2–3), the algorithm checks each vertex pair $\langle u, v\rangle$ for the occurrence of an edge (lines 4–7). If there is an edge, we assign its weight from mixture component m (lines 8,9). The generated graph is returned as $G = (V, E)$.

Algorithm 1 AGWAN Graph Generation

Require: N (no. of vertices), L (set of discrete label values), π (prior distribution over L), $\Theta = \{\Omega^{ij}\}$ (set of mixture models)

1: Create vertex set V of cardinality N, edge set $E = \emptyset$
2: **for all** $u \in V$ **do**
3: Randomly draw discrete label $l_u \in L$ according to prior π
4: **for all** $u, v \in V : u \neq v$ **do**
5: $i = l_u, j = l_v$
6: Randomly draw mixture component m according to mixture weights ω^{ij}
7: **if** $m \neq 0$ **then**
8: Randomly draw edge weight w_{uv} from $s^{ij} \cdot \mathrm{Beta}(a_m^{ij}, b_m^{ij})$
9: Create edge $e = \langle u, v, w_{uv}\rangle$, $E = E \cup \{e\}$
return $G = (V, E)$

3.2. Parameter Fitting

To create realistic random graphs, we need to learn the parameters π, Θ from a real-world input graph G. For each $i, j \in V$, the edge weights $W^{ij} = \{w_{ij}\}$ follow an unknown, arbitrary probability distribution. For each set of edge weights W^{ij}, we choose the scaling parameter $s^{ij} \geq \max_{W^{ij}}$, then estimate the BMM parameters for the empirical distribution $1/s^{ij} \cdot W^{ij}$, which has support $(0, 1]$.

In this section, we describe the Bayesian estimation of BMMs within a VI framework, followed by our algorithms for AGWAN parameter fitting.

The beta distribtion has a conjugate prior in the exponential family. The conjugate prior density function can be written as [15,28]:

$$f(a,b) = \frac{1}{C(\alpha_0,\beta_0,\nu_0)}\left[\frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)}\right]^{\nu_0} e^{-\alpha_0(a-1)}e^{-\beta_0(b-1)} \tag{3}$$

where α_0, β_0, ν_0 are free positive parameters and $C(\alpha_0,\beta_0,\nu_0)$ is a normalisation factor such that $\int_0^\infty\int_0^\infty f(a,b)dadb = 1$. Then, we obtain the posterior distribution of a, b as (with N independent and identically distributed (i.i.d.) scalar observations $X = \{x_1,\ldots,x_N\}$)

$$\begin{aligned} f(a,b|X) &= \frac{f(X|a,b)f(a,b)}{\int_0^\infty\int_0^\infty f(X|a,b)f(a,b)dadb} \\ &= \frac{1}{C(\alpha_N,\beta_N,\nu_N)}\left[\frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)}\right]^{\nu_N} e^{-\alpha_N(a-1)}e^{-\beta_N(b-1)}dadb \end{aligned} \tag{4}$$

where $\nu_N = \nu_0 + N$, $\alpha_N = \alpha_0 - \sum_{n=1}^N \ln x_n$ and $\beta_N = \beta_0 - \sum_{n=1}^N \ln(1-x_n)$. To avoid infinity in the practical implementation, we assign ε_1 to x_n when $x_n = 0$ and $1-\varepsilon_2$ to x_n when $x_n = 1$, where ε_1 and ε_2 are slightly positive real numbers.

We introduce an approximation to both the conjugate prior and the posterior distributions of the beta distribution and attempt to solve the Bayesian estimation problem via the factorised approximation method. A distribution can be used as the factorised distribution of the true posterior distribution if the optimal solution to this factorised distribution has exactly the same form as its initialisation. This requirement guarantees that the estimation works iteratively. With the non-negative property of the parameters in the beta distribution and assuming that a and b are statistically independent, we could use some well-defined distribution with support $(0,+\infty)$ to approximate the conjugate prior. One possible way is to assign the gamma distribution to a and b as:

$$f(a;\mu,\alpha) = \frac{\alpha^\mu}{\Gamma(\mu)}a^{\mu-1}e^{-\alpha a};\ f(b;\nu,\beta) = \frac{\beta^\nu}{\Gamma(\nu)}b^{\nu-1}e^{-\beta b} \tag{5}$$

The conjugate prior is then approximated as:

$$f(a,b) \approx f(a)f(b) \tag{6}$$

The same form of approximation applies to the posterior distribution as:

$$f(a,b|\mathbf{X}) \approx f(a|\mathbf{X})f(b|\mathbf{X}) \tag{7}$$

For each observation $\mathbf{x}_n$, the corresponding $\mathbf{z}_n = [z_{n1},\ldots,z_{nM}]^T$ is the indication vector defined with respect to the M components in the mixture model. One element of z_n will be equal to 1 and the rest equal to 0, to indicate which mixture component z_n belongs to. Denoting $Z = \{z_1,\ldots,z_N\}$ and assuming the indication vectors are independent given the mixing coefficients, the conditional distribution of Z given P is:

$$f(Z|P) = \prod_{n=1}^N\prod_{m=1}^M p_m^{z_{nm}} \tag{8}$$

Introducing the Dirichlet distribution as the prior distribution of the mixing coefficients, the probability function of P can be written as:

$$f(P) = Dir(p|c) = C(\mathbf{c})\prod_{m=1}^M p_m^{c_m-1} \tag{9}$$

where $C(\mathbf{c}) = \frac{\Gamma(\widehat{c})}{\Gamma(c_1)\cdots\Gamma(c_M)}$ and $\widehat{c} = \sum_{m=1}^{M} c_m$. We consider the observation $\mathbf{x}_n$ and the unobserved indication vector $\mathbf{z}_n$ as the *complete* data. The conditional distribution of $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ and $Z = \{\mathbf{z}_1, \ldots, \mathbf{z}_N\}$ given the latent variables $\{A, B, P\}$ is:

$$\begin{aligned} & f(\mathbf{X}, \mathbf{Z}|\mathbf{A}, \mathbf{B}, \mathbf{P}) \\ =& f(\mathbf{X}|\mathbf{A}, \mathbf{B}, \mathbf{P}, \mathbf{Z}) f(\mathbf{Z}|\mathbf{P}) \\ =& f(\mathbf{X}|\mathbf{A}, \mathbf{B}, \mathbf{Z}) f(\mathbf{Z}|\mathbf{P}) \\ =& \prod_{n=1}^{N} \prod_{m=1}^{M} [p_m \mathbf{Beta}(\mathbf{x}_n|\mathbf{a}_m, \mathbf{b}_m)]^{z_{nm}} \end{aligned} \tag{10}$$

The algorithm for Bayesian estimation of a BMM model is presented in Algorithm 2. An overview of the derivation of the algorithm from Equations (5)–(10) can be found in Appendix A. For full details of the derivations, refer to [26].

Algorithm 2 Bayesian estimation of a Beta Mixture Model (BMM)

Require: Number of components M, initial parameters for the Dirichlet distribution, initial parameters (element-wise) $\alpha_0 > 0$, $\beta_0 > 0$, $\mu_0 > 0.6156$, $\nu_0 > 0.6156$. Select initial parameters such that $\mu_0/\alpha_0 > 1$ and $\nu_0/\beta_0 > 1$.

1: Initialise r_{nm} with k-means
2: Calculate the initial guess of $\overline{a}$ and $\overline{v}$ from $\alpha_0, \beta_0, \mu_0, \nu_0$
3: **repeat**
4: Update the hyperparameters c_m^*, μ_{lm}^*, α_{lm}^*, ν_{lm}^* and β_{lm}^*
5: **until** convergence
return the current estimated hyperparameters

After initialisation (lines 1,2), Algorithm 2 iterates until the the current estimated model and the previous estimated model are sufficiently close (lines 3–5). The order of updating (line 4) does not matter, but each hyperparameter should be updated exactly once during each iteration. Refer to Appendix A for details of how the intermediate quantities are calculated (c_m^* is updated following Equation (A3), μ_{lm}^* from Equation (A4), α_{lm}^* from Equation (A5), ν_{lm}^* from Equation (A6) and β_{lm}^* from Equation (A7). The expectations for these quantities are given in Equation (A8).) The algorithm returns the current estimated hyperparameters at the last iteration, which are used to get the approximating posterior distribution. The joint posterior distribution of a_{lm}, b_{lm} (Equation (4)) is approximated by the product of two gamma distributions with parameters $\mu_{lm}^*, \alpha_{lm}^*$ and ν_{lm}^*, β_{lm}^* (Equations (5) and (7)).

AGWAN parameter fitting is performed by Algorithm 3. First, we estimate the vertex label priors (lines 1–3); Next, we sample the edge weights for each possible combination of vertex label values (lines 5–10); The proportion of non-edges is used to estimate the weight of mixture 0 (line 10). We estimate each scaled BMM Ω^{ij} from the appropriate set of samples W^{ij} using Algorithm 2 as described above (lines 12–13). Finally, the mixture weights ω_m^{ij} are normalised so that they sum to 1 (line 14).

Algorithm 3 AGWAN Parameter Fitting

Require: Input graph $G = (V, E)$

$L = \{\text{discrete vertex label values}\}$, $d = |L|$
Calculate vertex label priors, apply Laplace smoothing $\forall l \in L : P(l) = \frac{count(l)+\alpha}{N+\alpha d}$
$\pi =$ the normalised probability distribution over L such that $\sum_{i=1}^{d} P(l_i) = 1$
$\forall i, j \in L : W^{ij} = \emptyset$
for all $u, v \in V : u \neq v$ **do**
    $i = l_u, j = l_v$
    if $\langle u, v \rangle \in E$ **then**
        $W^{ij} = W^{ij} \cup \{w_{uv}\}$
    else
        Increment ω_0^{ij}
for all $i, j \in L$ **do**
    $\omega_0^{ij} = 1 - P(e_{ij}$ exists$)$
    $s^{ij} = \max_{W^{ij}}$
    estimate Ω^{ij} from $1/s^{ij} \cdot W^{ij}$ using Algorithm 2
    Normalise all ω_m^{ij}
    return $\pi, \Theta = \{\Omega^{ij}\}$

3.3. Extending AGWAN *to Multiple Attributes*

AGWAN can be extended to multiple numeric edge labels by generalising the concept of edge weight to K dimensions. In this case, the weight is parameterised as the multivariate beta distribution. For any random vector x consisting of K elements, the dependencies among the elements $x_1, \ldots, x_K$ can be represented by a mixture model, even if each specific mixture component can only generate vectors with statistically independent elements. Therefore, we define the multivariate BMM as:

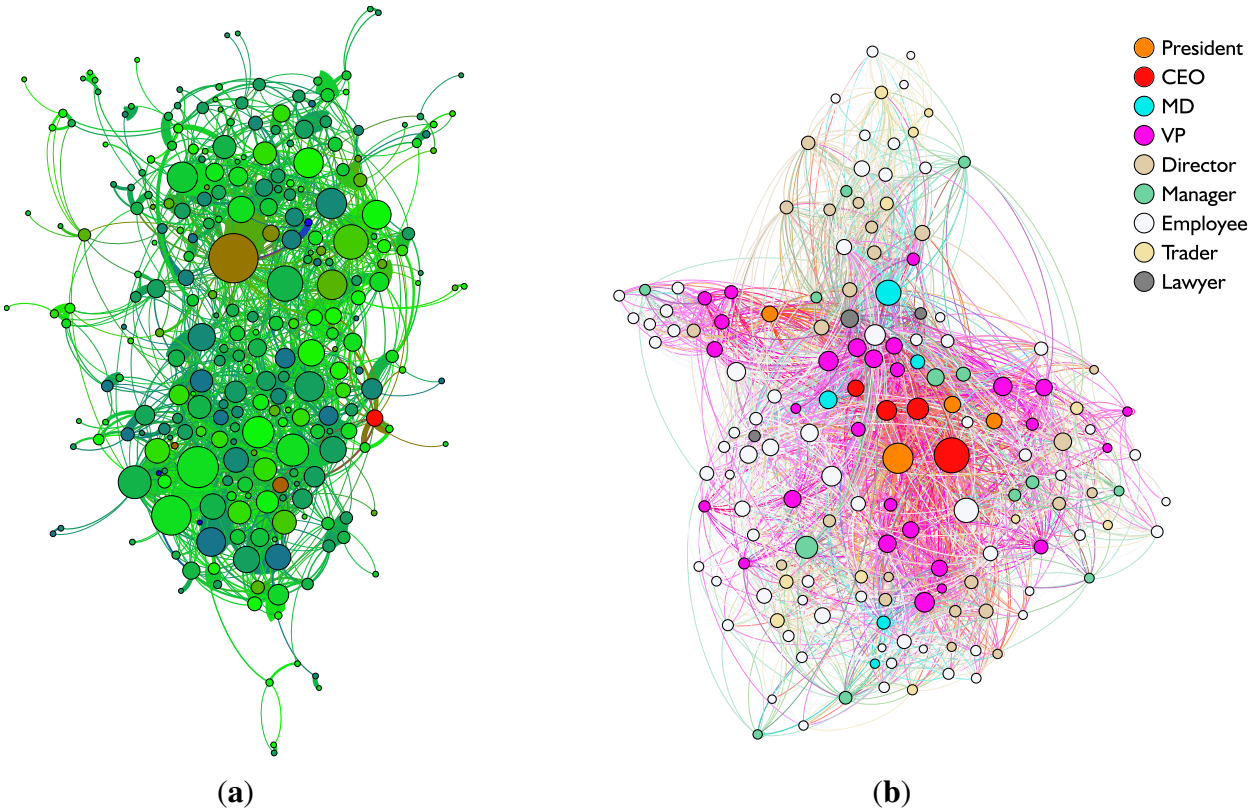
$$f(x; P, A, B) = \sum_{m=1}^{M} p_m \cdot \mathbf{Beta}(x; a_m, b_m) = \sum_{m=1}^{M} p_m \cdot \prod_{k=1}^{K} \text{Beta}(x_k; a_{km}, b_{km}) \tag{11}$$

where $x = \{x_1, \ldots, x_K\}$, $P = \{p_1, \ldots, p_M\}$, $A = \{a_1, \ldots, a_M\}$ and $B = \{b_1, \ldots, b_M\}$. a_m, b_m denote the parameter vectors of the m^{th} mixture component and a_{km}, b_{km} are the (scalar) parameters of the beta distribution for element x_k. Using this representation, we can apply Algorithm 2 to K-dimensional edge weights.

4. Experimental Section

We evaluate our approach by comparing AGWAN with the state-of-the-art in labelled graph generation, represented by the MAG model [9,14]. AGWAN and MAG parameters are learned from real-world graphs. We generate random graphs from each model and calculate a series of statistics on each graph. As MAG does not generate weighted graphs, we fixed the weight of the edges in the generated graphs to the mean edge weight from the input graphs. This ensures that statistics such as average vertex strength are maintained.

We used two datasets for the experiments. The first is a graph of "who exercises with whom" from a behavioural health study (Figure 2a, $|V| = 279, |E| = 1308$) [29]. Vertices represent participants and are labelled with 28 attributes denoting demographic information and health markers obtained from questionnaire data. Participants were tracked during periods of physical activity; when two people frequently register at the same sensor at the same time, an undirected edge is created, weighted with the number of mutual coincidences. Our second dataset is the "who communicates with whom" graph of the Enron e-mail corpus (Figure 2b, $|V| = 159, |E| = 2667$) [11]. Vertices are labelled with the job role of the employee. Edges are weighted with the number of e-mails exchanged between sender and recipient. As e-mail communications are not symmetric, edges in the Enron graph are directed. Both graphs exhibit a core-periphery structure which is typical of many real-world networks [7].



(**a**) (**b**)

Figure 2. Input Graph Datasets, from (**a**) a health study and (**b**) the Enron e-mail corpus. (**a**) Undirected graph of who exercised with whom; (**b**) Directed graph of who e-mailed whom.

We evaluated AGWAN against the following models:

Erdős-Rényi random graph (ER): The ER model $G(n, p)$ has two parameters. We set the number of vertices n and the edge probability p to match the input graphs as closely as possible. We do not expect a very close fit, but the ER model provides a useful baseline.

MAG model with real attributes (MAG-R1) The MAG model with a single real attribute has a set of binary edge probabilities, $\Theta = \{p^{ij}\}$ instead of a set of BMMs $\Theta = \{\Omega^{ij}\}$.

MAG model with latent attributes (MAG-Lx): The MAG model also allows for modelling the graph structure using latent attributes. The discrete labels provided in the input graph are ignored; instead

MAGFIT [14] learns the values of a set of latent attributes to describe the graph structure. We vary the number of synthetic attributes x to investigate the relative contributions of vertex labels and edge weights to graph structure.

AGWAN model with truncated GMMs: We compare our BMM model to an alternative approach using GMMs [20]. One drawback of using GMMs is that it is possible to draw edge weights $w_{uv} < 0$. To avoid negative edge weights, we implement a *tabula rasa* rule during graph generation, drawing new values from Ω^{ij} until $w_{uv} \geq 0$.

To evaluate the closeness of fit of each model, we use the following statistics:

Vertex Strength: For an unweighted graph, one of the most important measures is the degree distribution (the number of in-edges and out-edges of each vertex). Real-world graphs tend to have heavy-tailed power-law or log-normal degree distributions [2,5]. For a weighted graph, we generalise the concept of vertex degree to vertex strength [30]:

$$s_u = \sum_{v \neq u} w_{uv} \tag{12}$$

We compare using the Complementary Cumulative Distribution Function (CCDF) of the strength of each vertex (both in-strength and out-strength in the case of the directed graph). For Cumulative Distribution Function (CDF) $F(x) = P(X \leq x)$, the CCDF is defined as $\bar{F} = P(X > x) = 1 - F(x)$. We show the unnormalised CCDFs in our plots; the normalised value can be obtained by integrating the area under the curve to 1. The CCDF of a power-law function will appear linear when plotted on a log-log scale, while the CCDF of a log-normal function will appear parabolic.

Spectral Properties: We use Singular Value Decomposition (SVD) to calculate the singular values and singular vectors of the graph's adjacency matrix, which act as a signature of the graph structure. In an unweighted graph, the adjacency matrix contains binary values, for "edge" or "no edge". In a weighted graph, the adjacency matrix contains the edge weights (with 0 indicating no edge). For SVD $U\Sigma V$, we plot the CCDFs of the singular values Σ and the components of the left singular vector U corresponding to the highest singular value.

Clustering Coefficients: the clustering coefficient C is an important measure of community structure. It measures the density of triangles in the graph, or the probability that two neighbours of a vertex are themselves neighbours [5]. We extend the notion of clustering coefficients to weighted, directed graphs by defining C_u, the weighted clustering coefficient for vertex u [30]:

$$C_u = \frac{[\mathbf{W}_u^{[\frac{1}{3}]} + (\mathbf{W}_u^T)^{[\frac{1}{3}]}]_{uu}^3}{2[d_u^{tot}(d_u^{tot} - 1) - 2d_u^{\leftrightarrow}]} \tag{13}$$

where $\mathbf{W}_u$ is the weighted adjacency matrix for u and its neighbours, $\mathbf{W}^T$ is the transpose of $\mathbf{W}$, d_u^{tot} is the total degree of a vertex (the sum of its in- and out-degrees) and $d_u^{\leftrightarrow}$ is the number of bilateral edges in u (the number of neighbours of u which have both an in-edge and an out-edge between themselves and u). The notation $\mathbf{A}_{uu}$ means the u^{th} element of the main diagonal of $\mathbf{A}$.

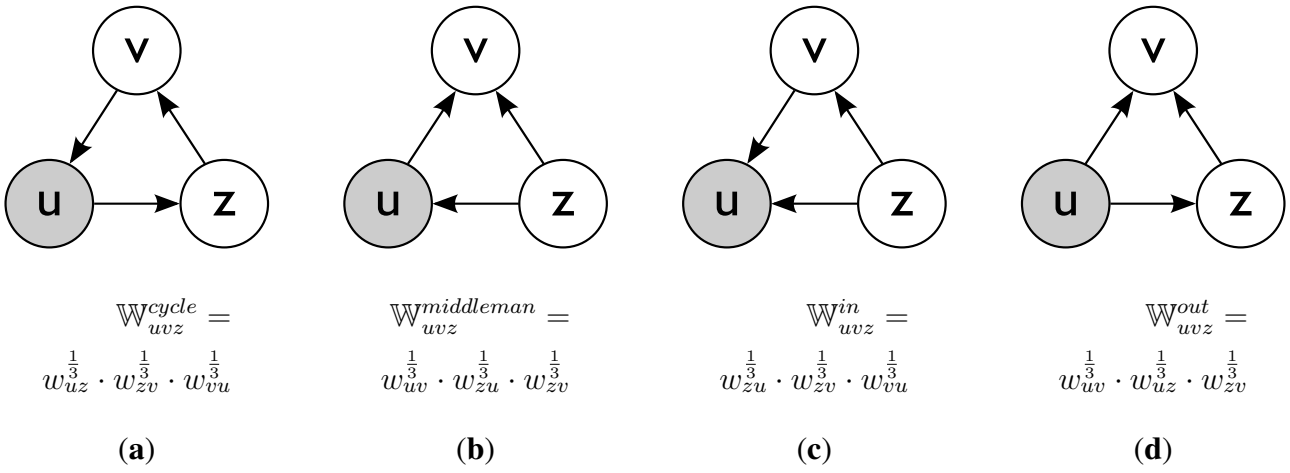


Figure 3. Triad Patterns in a Directed Graph. (**a**) Cycle; (**b**) Middleman; (**c**) In; (**d**) Out.

Triad Participation: Closely related to the clustering coefficient is the concept of triangle or triad participation. The number of triangles that a vertex is connected to is a measure of transitivity [5]. For the directed graphs, the triangles have a different interpretation depending on the edge directions. There are four types of triangle pattern [30], as shown in Figure 3. To generalise the concept of triad participation to weighted, directed graphs, we consider each of the four triangle types separately, and sum the total strength of the edges in each triad:

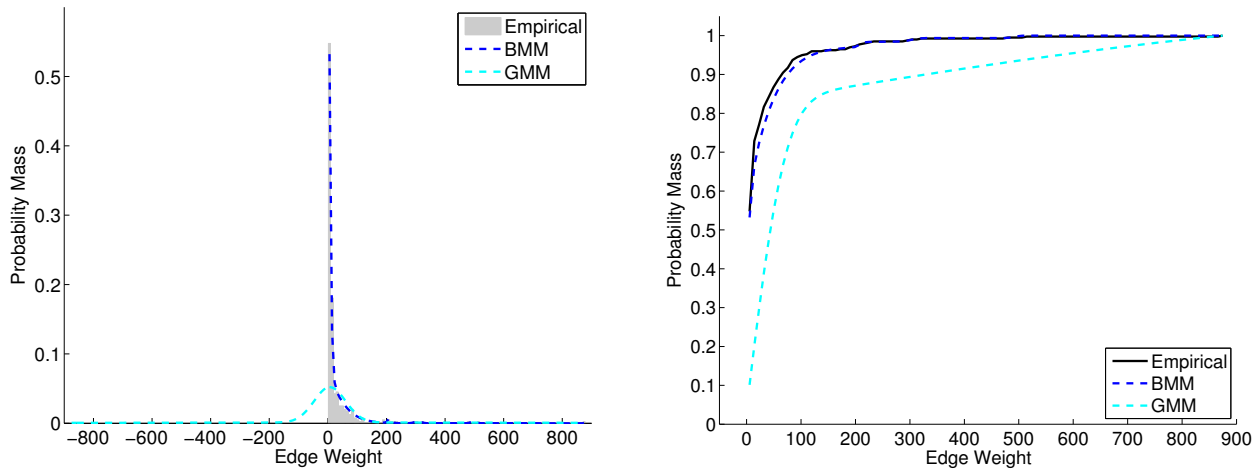
$$t^y_u = \sum_{v,z \in \mathbf{W}_u \setminus u} \mathbb{W}^y_{uvz} \tag{14}$$

where $y = \{cycle, middleman, in, out\}$ is the triangle type and $\mathbb{W}^y_{uvz}$ is calculated as shown in Figure 3 for each triangle type y.

To give a more objective measure of the closeness of fit between the generated graphs and the input graph, we use a Kolmogorov-Smirnov (KS) test and the L2 (Euclidean) distance between the CCDFs for each statistic. Details are in Appendix B.

5. Results and Discussion

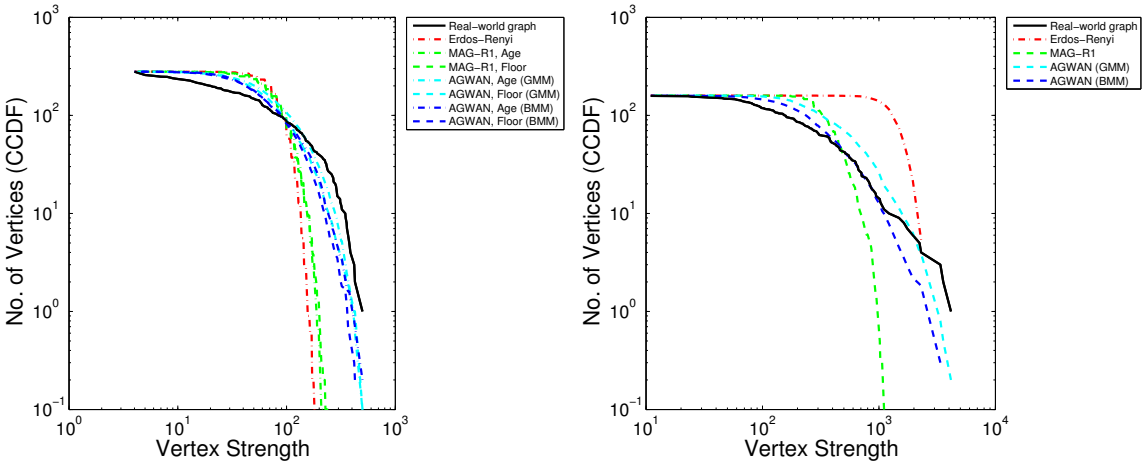
In the experimental data, most of the edge weights follow a heavy-tailed distribution. The BMM achieves a very close fit with its primary mixture component (Figure 4a). The GMM would need several Gaussian components to achieve a similar fit. In practice the VI algorithm for GMMs [18] tries to fit to power law or log-normal distributions using a single Gaussian component, resulting in probability mass being assigned to the area $x < 0$, as shown in Figure 4a. This results in a fit that is not as close as the BMM (Figure 4b). To compensate for this effect, we used a truncated GMM for graph generation as discussed in Section 3.2.



(**a**) (**b**)

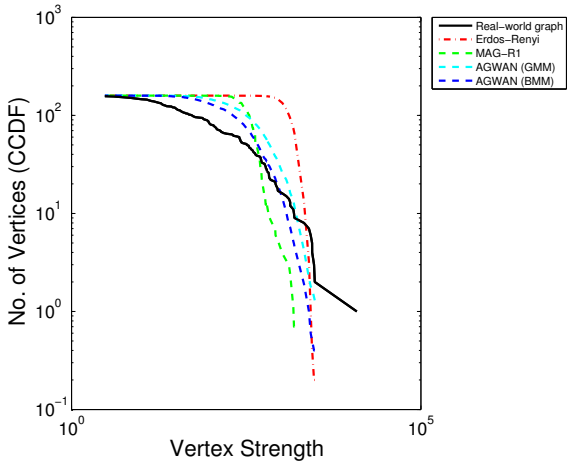
Figure 4. Fitting BMM and GMM models to edge weights. (**a**) Probability Mass Function; (**b**) Cumulative Distribution Function.

For our experiments, we generated 10 random graphs for each AGWAN model. For each graph, we calculated the statistics for vertex strength, spectral properties, clustering and triad participation, as described in the previous section. We calculated the CCDFs for each set of statistics, and averaged the CCDF scores at each x-coordinate value across the 10 graphs, to smooth any random fluctuations. The plots of the averaged CCDFs for each model are shown in Figures 5–12. Tables for the closeness of fit of each CCDF (KS and L2 statistics) are in Appendix B.



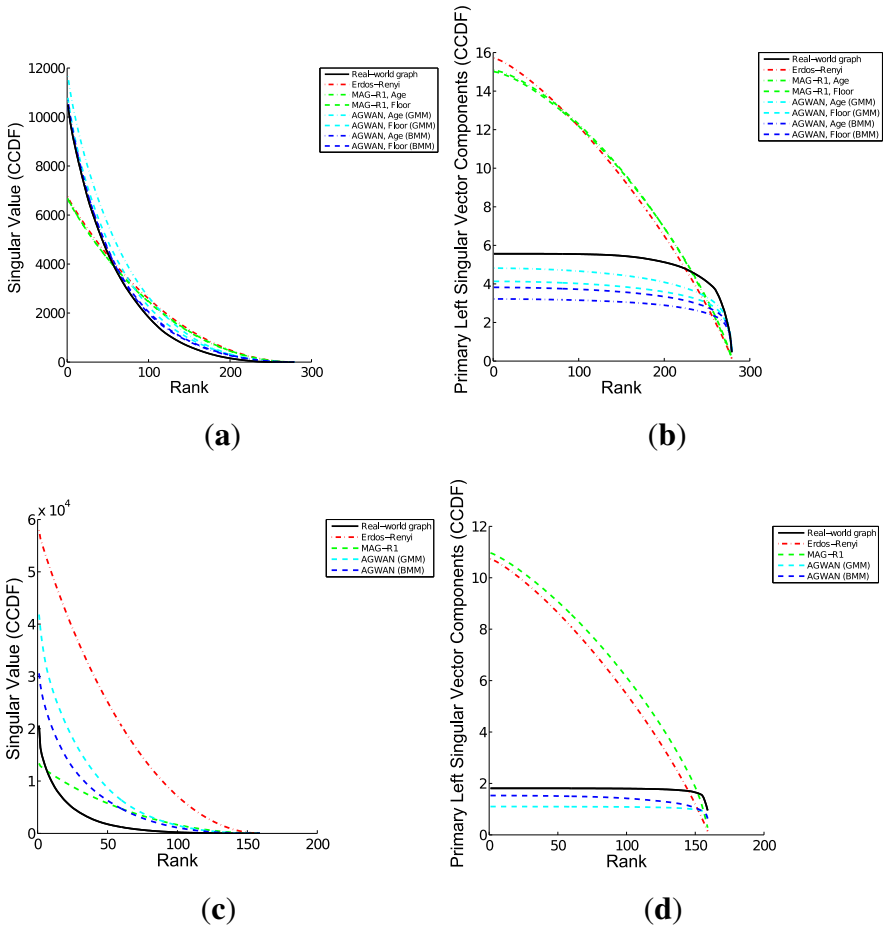
(**a**) (**b**)

Figure 5. *Cont.*



(**c**)

Figure 5. Vertex Strength Distribution—Real Attributes. (**a**) Undirected; (**b**) Directed (In-strength); (**c**) Directed (Out-strength).



(**a**) (**b**)

(**c**) (**d**)

Figure 6. Spectral Properties—Real Attributes. (**a**) Undirected—Singular Values; (**b**) Undirected—Primary Left Singular Vector; (**c**) Directed—Singular Values; (**d**) Directed—Primary Left Singular Vector.

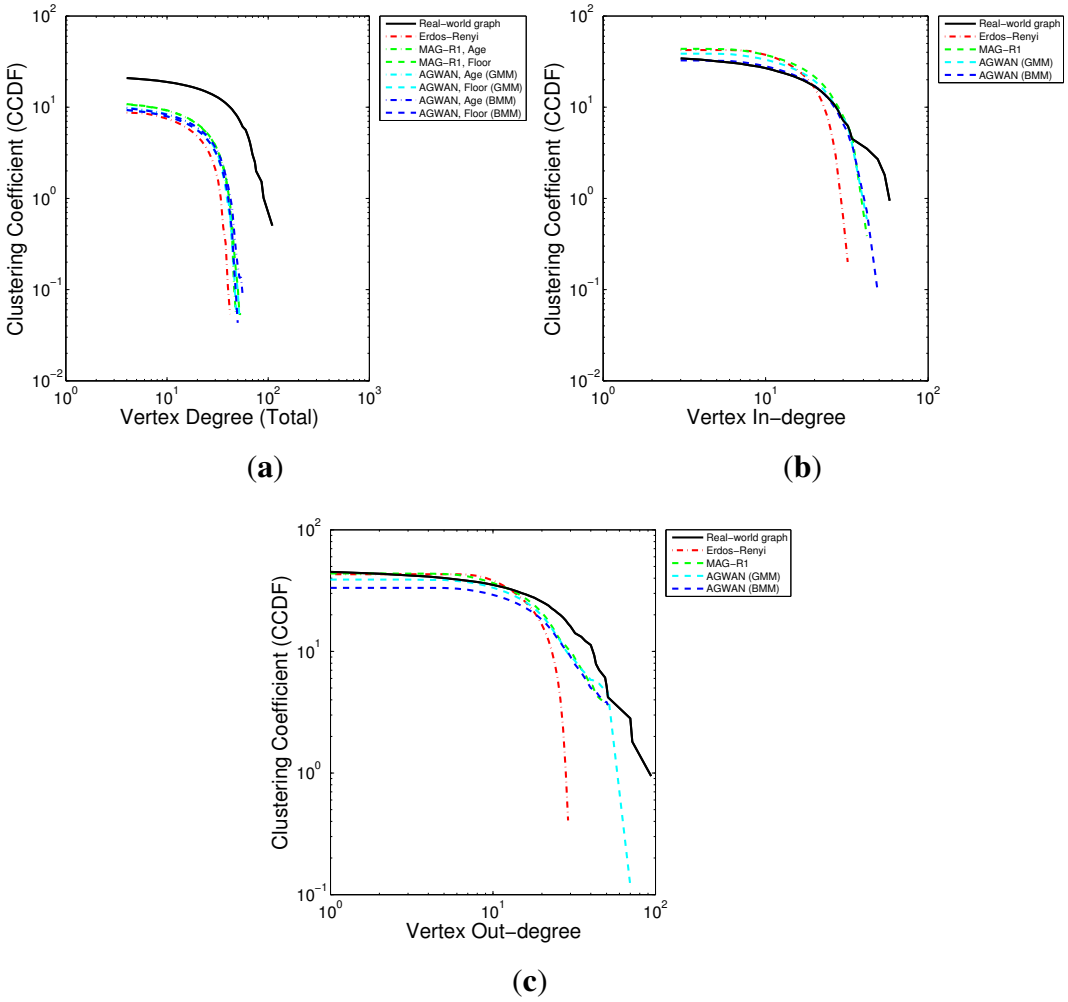


Figure 7. Clustering Coefficients—Real Attributes. (**a**) Undirected; (**b**) Directed (In-edges); (**c**) Directed (Out-edges).

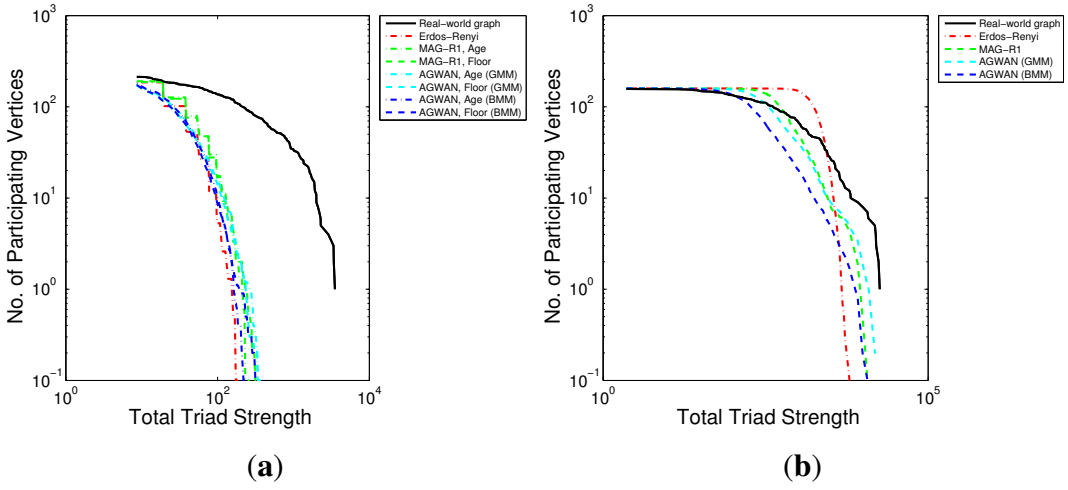


Figure 8. *Cont.*

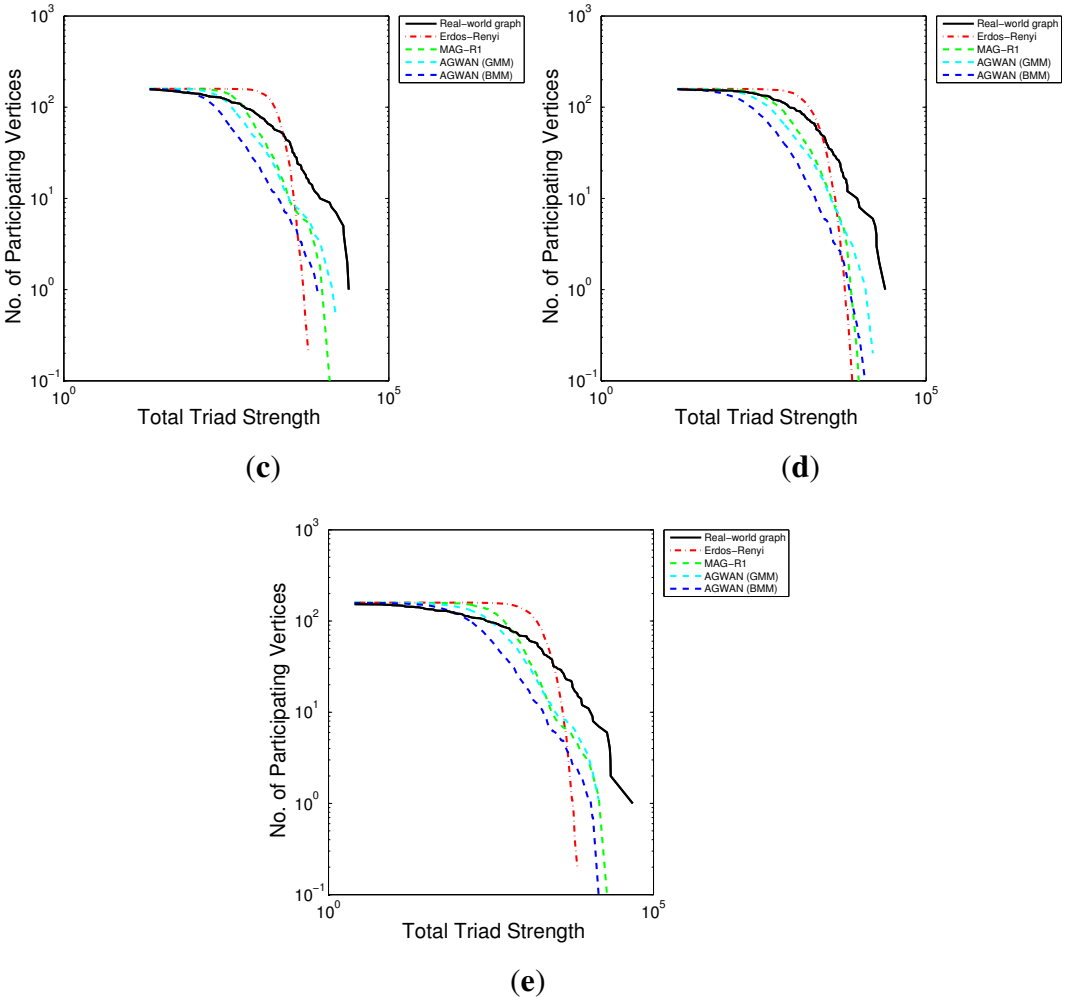


Figure 8. Triad Participation—Real Attributes. (**a**) Undirected; (**b**) Directed (Cycles); (**c**) Directed (Middlemen); (**d**) Directed (Ins); (**e**) Directed (Outs).

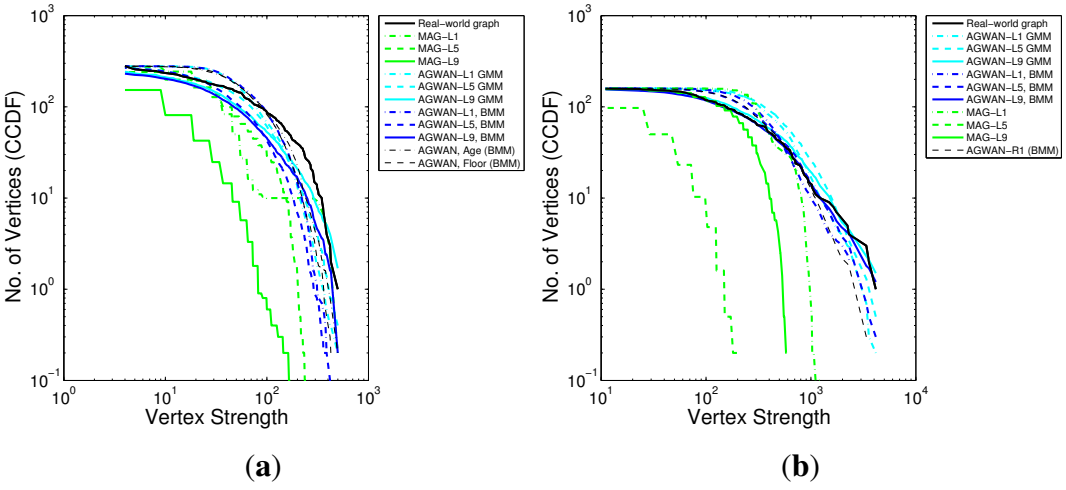
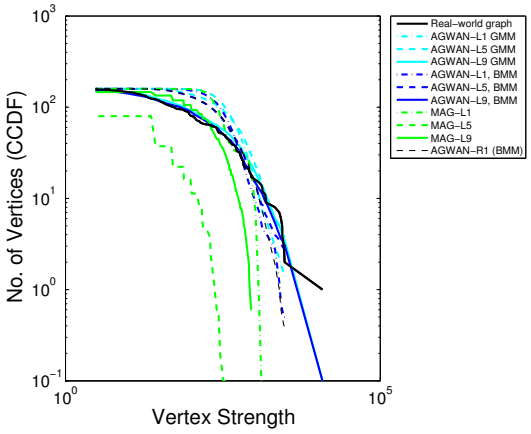
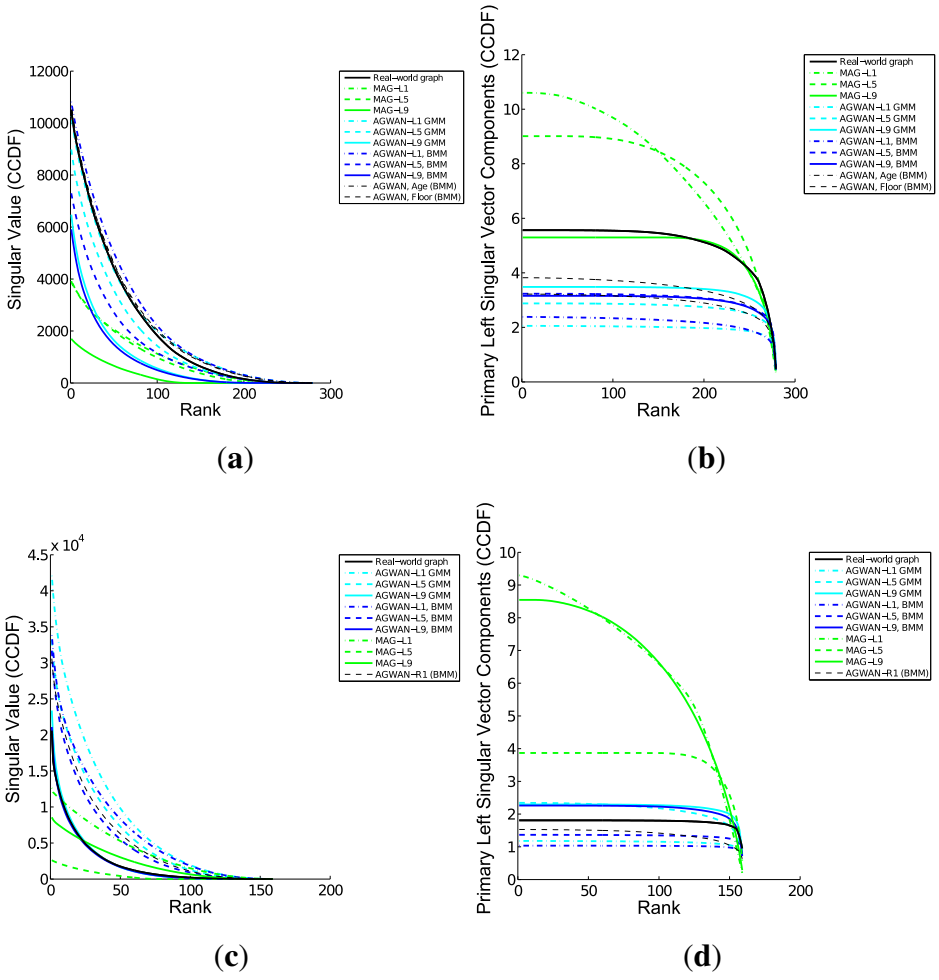


Figure 9. *Cont.*



(**c**)

Figure 9. Vertex Strength Distribution—Synthetic Attributes. (**a**) Undirected; (**b**) Directed (In-strength); (**c**) Directed (Out-strength).



(**a**) (**b**)

(**c**) (**d**)

Figure 10. Spectral Properties—Synthetic Attributes. (**a**) Undirected—Singular Values; (**b**) Undirected—Primary Left Singular Vector; (**c**) Directed—Singular Values; (**d**) Directed—Primary Left Singular Vector.

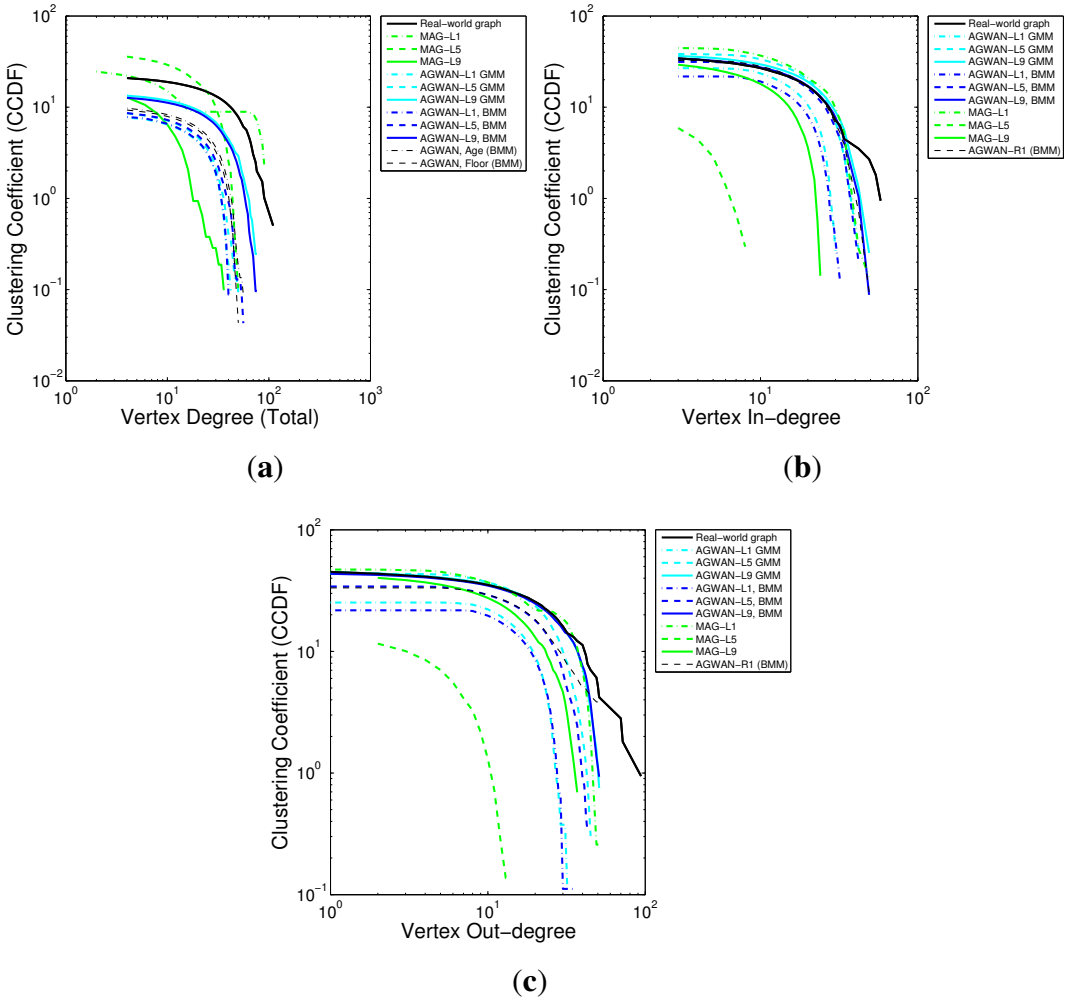


Figure 11. Clustering Coefficients—Synthetic Attributes. (**a**) Undirected; (**b**) Directed (In-edges); (**c**) Directed (Out-edges).

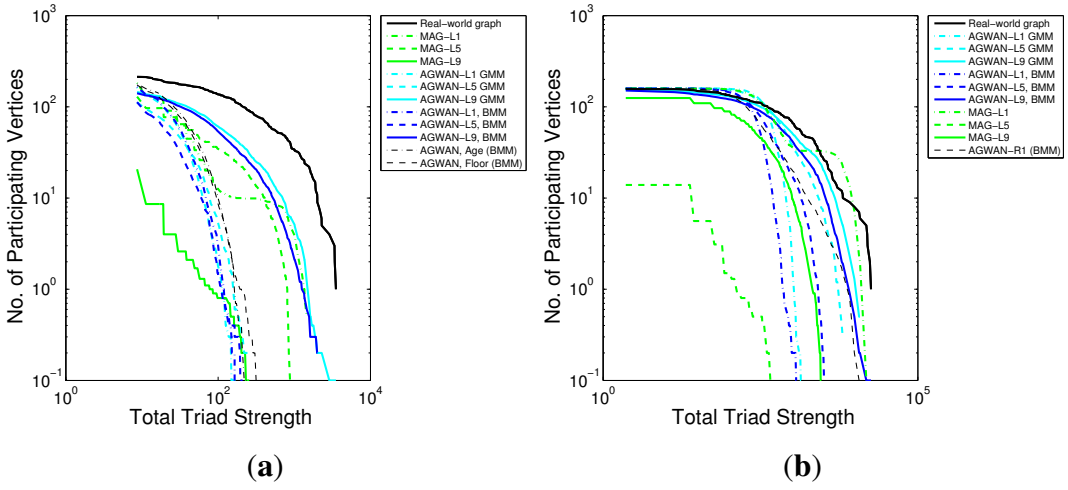


Figure 12. *Cont.*

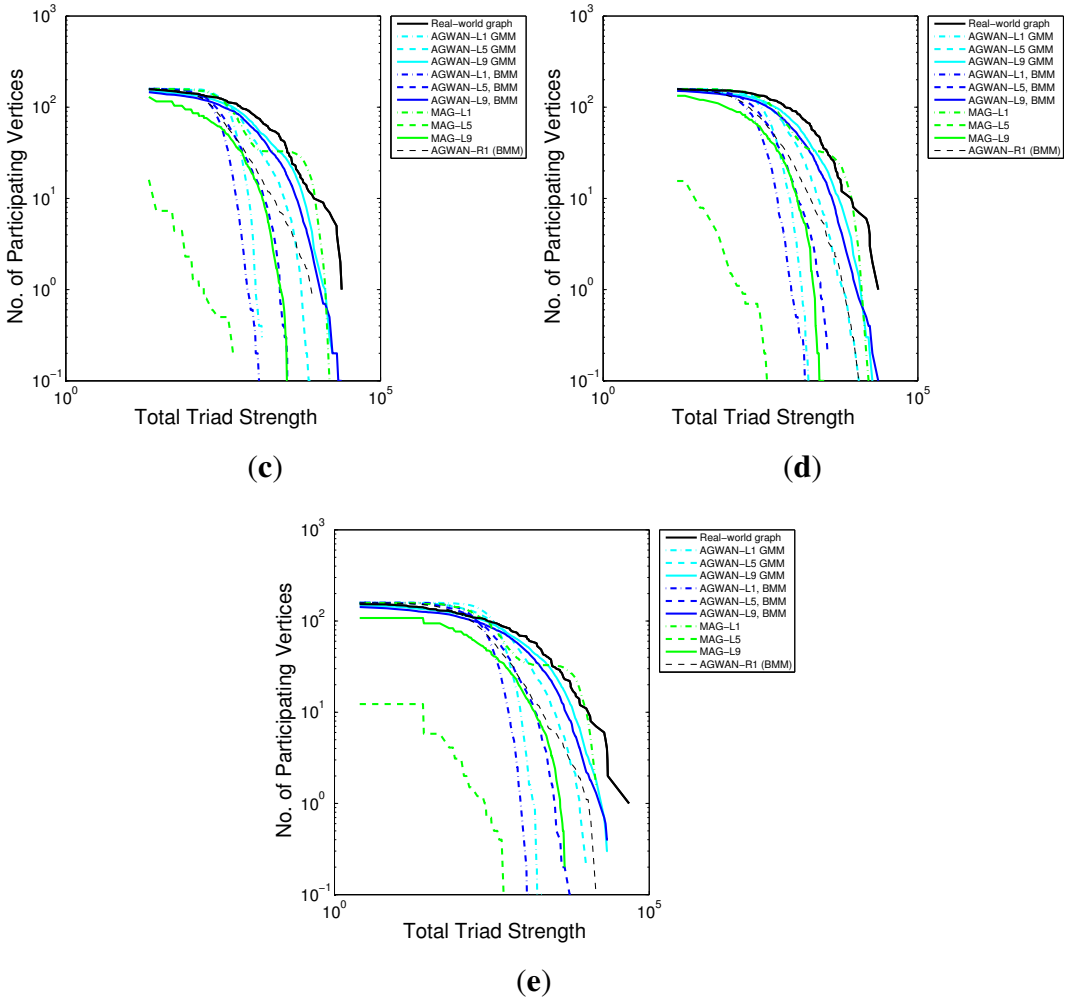


Figure 12. Triad Participation—Synthetic Attributes. (**a**) Undirected; (**b**) Directed (Cycles); (**c**) Directed (Middlemen); (**d**) Directed (Ins); (**e**) Directed (Outs).

5.1. Real Attributes

For the undirected graph (Health Study), we show results for two vertex attributes: Age and Floor (the building and floor number where the person works; people who work on the same floor were highly likely to exercise together). For the directed graph (Enron), there is one vertex attribute, the person's job role.

Vertex Strength (Figure 5): The graphs generated from AGWAN have vertex strength distributions which map very closely to the input graphs. The graphs generated from MAG-R1 are better than random (ER), but the vertex strength distribution is compressed into the middle part of the range, with too few high- and low-strength vertices. This indicates that vertex strength depends on both the label distribution and the edge weight distribution; AGWAN models both of these effectively.

Spectral Properties (Figure 6): The spectral properties of the AGWAN graphs map very closely to the input graphs. The singular values follow the same curve as the input graphs. This is significant as it indicates that graphs generated with AGWAN have similar connectivity to the input graph: the primary singular value has been shown to be the most significant measure of graph connectivity [2]. In contrast,

MAG-R1 does not preserve the singular value curve, showing a linear relationship instead. MAG-R1 cannot model the singular vector components at all without taking the edge weights into account. These results show that the spectral properties of the graph are dependent on both the vertex label distribution and the edge weight distribution.

Clustering Coefficients (Figure 7): The accuracy of AGWAN and MAG-R1 is similar. For the undirected graph, performance is little better than random. For the directed graph, AGWAN gives reasonable performance for the low-degree vertices but drops away for the high-degree vertices. In all cases, the clustering is independent of the edge weight distribution. The accuracy of the results depends on which vertex attribute was chosen, implying that some attributes can predict community formation better than others. We hypothesise that cluster formation may in fact be (conditionally) independent of the vertex label values and may be better explained by triadic closure [5]: links are likely to form between two people who share a mutual friend, independently from their vertex attributes. The apparent dependency on vertex labels may be an artefact of the connection to the mutual friend, rather than the true explanation of why clustering asises. This aspect of network formation requires further investigation.

Triad Participation (Figure 8): Similar to clustering, triad participation appears to be dependent to some extent on vertex label values but independent of the edge weight distribution. We hypothesise that like clustering, the apparent dependency between vertex label and triad participation values may be due to triadic closure, which is not currently modelled by either MAG or AGWAN.

5.2. Synthetic Attributes

The second set of experiments ignore the real attributes of the graph, replacing them with a set of randomly generated synthetic attributes. The purpose is to evaluate the relative contribution of the discrete vertex labels and numeric attributes to the graph structure. By reducing the number of numeric attributes to zero, we can evaluate the contribution of the numeric attributes in isolation.

We replaced the real labels in the input graph with a synthetic vertex attribute taking $2^0 \ldots 2^9$ values allocated uniformly at random, then learned the edge weight distributions using VI as normal. We compare our results with an alternate interpretation of the MAG model, which ignores the true attribute values from the input graph and represents attributes as latent variables, which are learned using a VI EM approach [14]. We also show AGWAN with one real attribute for comparison.

Vertex Strength (Figure 9): AGWAN with synthetic attributes significantly outperforms MAG for the accuracy of the vertex strength distribution, and has similar accuracy to AGWAN-R1. Varying the number of synthetic attributes has a small effect on the accuracy. We conclude that vertex strength is dependent on both edge weight and vertex label distribution, but the edge weights play a more important role.

Spectral Properties (Figure 10): AGWAN has a very close fit to the spectral properties of the input graphs. Varying the number of attributes has a moderate effect on the closeness of fit. On the undirected graph, MAG-L9 matches the singular vector very closely but performs poorly with the singular values; in general, MAG is a poor fit as the edge weight distribution is not taken into account. These results confirm that the spectral properties are dependent on both the vertex label distribution and the edge weight distribution, and the degrees of freedom of the vertex labels are important.

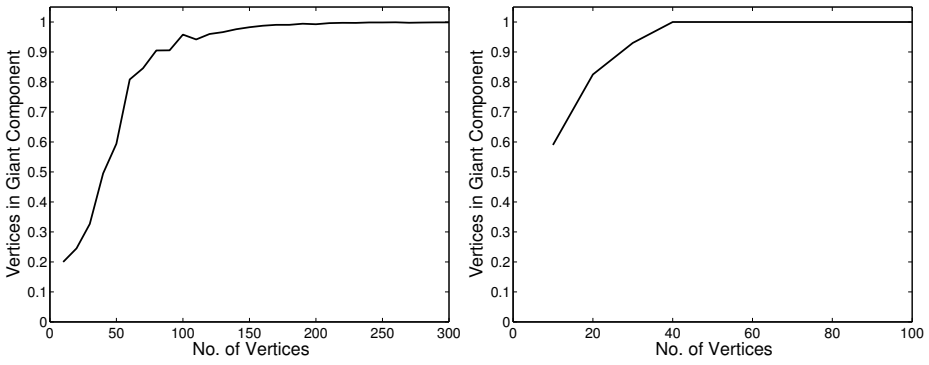
Clustering Coefficients (Figure 11): AGWAN and MAG are both significantly more accurate using synthetic attributes than with real attributes. This (somewhat surprising) result implies that clustering is not closely related to the real attribute values. While real labels appear to be influenced by the (unobserved) process which gives rise to clustering, synthetic labels with more degrees of freedom can model it more accurately. We note that AGWAN performs better where there are more degrees of freedom, while MAG performs better with few degrees of freedom (due to the independence assumption mentioned in Section 3.3). As before, clustering appears to be independent of the edge weight distribution.

Triad Participation (Figure 12): Similarly to clustering, AGWAN and MAG are both more accurate using synthetic attributes than they were with real attributes. The effects of degrees of freedom of the synthetic attributes is even more pronounced than it was for clustering: AGWAN-L9 has a good fit whereas AGWAN-L1 is not so good. Edge weights appear to have little influence.

5.3. Graph Evolution

One of the goals of our model is to generate synthetic graphs which are larger than the input graph. We conducted experiments to measure how AGWAN graph properties change as the number of vertices is increased. We generated sets of random graphs from the same model, with increasing numbers of vertices $N = \{10, 20, \ldots, 100, 200, \ldots, 3000\}$ vertices and measured the size of the giant component, the density and the diameter.

Giant Component: We measured the proportion of vertices in the largest component of the graph as the graph grows in size. Figure 13 shows that a giant component forms quickly, as in real graphs [5].



(**a**) (**b**)

Figure 13. Giant Component. (**a**) Undirected; (**b**) Directed.

Densification Power Law: We measured graph densification as the ratio of the number of vertices to the number of edges as the graph grows in size. Figure 14 shows that AGWAN graphs densify as they grow, as in real graphs. Real-world graphs have $e(t) \propto v(t)^{\gamma}$, with power-log exponent γ typically in the range $(1, 2)$ [4]. When $\gamma = 1$, the rate of edge growth is linear in the number of vertices; $\gamma = 2$ means that on average, each vertex has edges to a constant fraction of all vertices. As Figure 14 shows, AGWAN graphs have $\gamma \simeq 2$, because the model learns its edge probabilities independently from the size

of the input graph. This means that AGWAN overestimates density when generating graphs larger than the input graph.

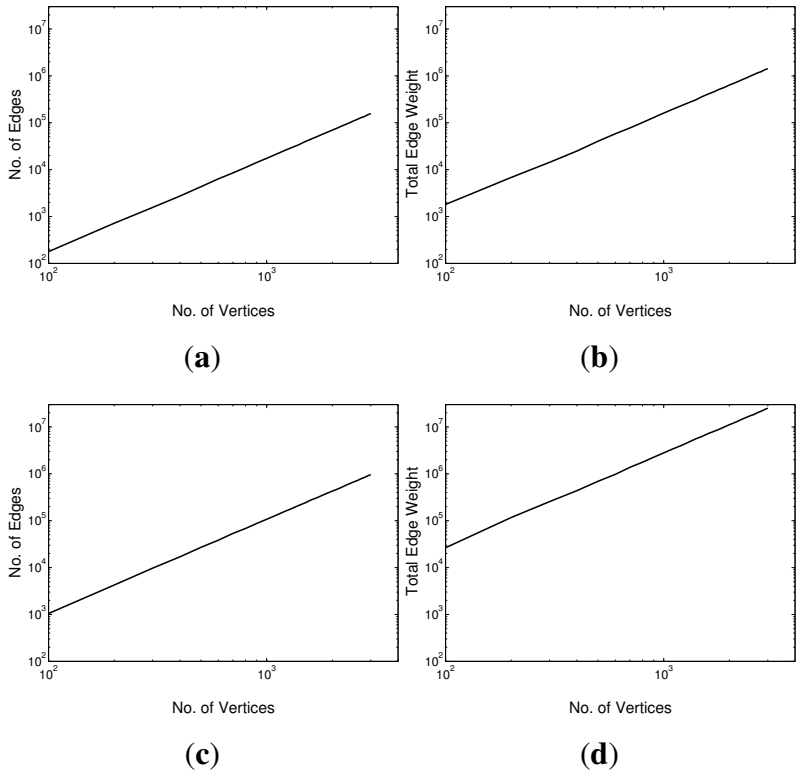


Figure 14. Densification Power Law. (**a**) Undirected, unweighted; (**b**) Undirected, weighted; (**c**) Directed, unweighted; (**d**) Directed, weighted.

Shrinking Diameter Effect (Figure 15): AGWAN graphs exhibit the Small World Effect [5] and the unweighted diameter shrinks as the graphs grow, as in real graphs [4]. As the weighted diameter was calculated using Johnson's algorithm [31], which treats the edge weight as a cost function, we used the reciprocal of the edge weights. With this interpretation, the weighted diameter also shrinks as the graphs grow. It is interesting to note that the weighted diameters decay following a power law (with exponent $\gamma = -0.72$ for the undirected graph and $\gamma = -0.65$ for the directed graph).

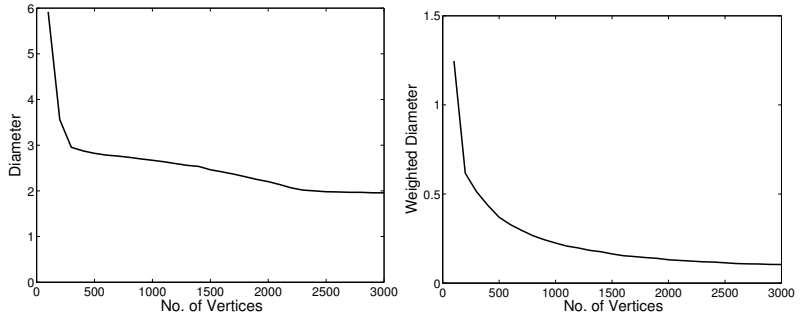
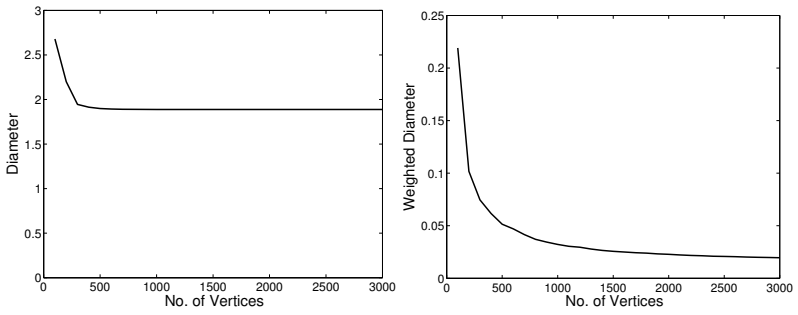


Figure 15. *Cont.*



(**c**) (**d**)

Figure 15. Shrinking Diameter Effect. (**a**) Undirected, unweighted; (**b**) Undirected, weighted; (**c**) Directed, unweighted; (**d**) Directed, weighted.

6. Conclusions and Future Work

In this paper, we presented AGWAN, a generative model for random graphs with discrete labels and weighted edges. We included a fitting algorithm to learn a model of edge weights from real-world graphs, and an algorithm to generate random labelled, weighted graphs with similar characteristics to the input graph.

We measured the closeness of fit of our generated graphs to the input graph over a range of graph statistics, and compared our approach to the state-of-the-art in random graph generative algorithms. Our results and findings are summarised in Table 1.

Table 1. Summary of results and findings: dependencies between graph labels and weights and structural properties (*hypothesised); relative accuracy of the properties of weighted graphs generated using AGWAN and MAG models.

Statistic	Vertex Labels	Edge Weights	AGWAN	MAG
Vertex Strength	Dependent	Dependent	Accurate	Less accurate
Singular Values	Dependent	Dependent	Accurate	Less accurate
Primary Singular Vector	Partially Dependent	Dependent	Accurate	Poor
Clustering Coefficients	Conditionally Independent*	Independent	Less accurate when using real attributes More accurate with synthetic attributes	
Triad Participation	Conditionally Independent*	Independent	Less accurate when using real attributes More accurate with synthetic attributes	

Our results demonstrate that AGWAN produces an accurate model of the weighted input graphs. The AGWAN graphs reproduce many of the properties of the input graphs, including formation of a giant component, heavy-tailed degree distribution, Small World property, Densification Power Law and shrinking diameter effect. Vertex strength and spectral properties were modelled very closely, indicating that these properties are indeed dependent on vertex labels and edge weights. For clustering and triad participation, it appears that these properties are independent of the edge weight distribution. While there

appears to be a relationship with the vertex label distribution, we suggest that this may be an artefact of the true process giving rise to these properties, triadic closure. Further research is required into the relationship between vertex attributes and triangle formation in graphs.

We investigated the accuracy of modelling edge weights using BMMs and compared to a previous approach which used GMMs. The edge weights often follow a power-law distribution, which the BMM approach fits more closely, particularly in the lower range of edge weight values. In general, BMMs are more suitable for bounded data; probability mass can only be assigned to the valid range of weight values. For the GMM model, we compensated for this by truncating the GMM during graph generation. We expected that AGWAN (BMM) would consistently outperform AGWAN (GMM), but the truncated GMM performed surprisingly well. We propose to conduct experiments on a wider range of datasets to investigate this further. It would also be interesting to compare with Poisson Mixture Models to model discrete edge weights.

Supplementary Materials

Source code and datasets used for the experiments (Gzipped TAR archive). See `Datasets/README` and `src/README` for details. Supplementary materials can be accessed at: http://www.mdpi.com/1999-4893/8/4/1143/s1.

Acknowledgments

This research was supported by funding from the Engineering and Physical Sciences Research Council (grant No. EP/H049606/1), National Nature Science Foundation of China (NSFC) grant Nos. 61402047 and 61511130081, the National Prevention Research Initiative (grant No. G0802045) and their funding partners, and the Department for Employment and Learning (grant No. M6003CPH).

Author Contributions

Michael Charles Davis is the principal author, responsible for the design of the study, conducting the experiments and analysing the results. Zhanyu Ma contributed the variational inference approach used for Beta Mixture Modelling. Weiru Liu and Paul Miller supervised the study and provided substantial critique and guidance. Ruth Hunter and Frank Kee designed the social network study for the health intervention and provided data for the experiments.

Conflicts of Interest

The authors declare no conflict of interest.

Appendix

A. Derivation of Approximate Inference Algorithm

The derivation of the algorithm for Bayesian estimation of a BMM model (Algorithm 2) from Equations (5)–(10) is outlined in this appendix.

We combine Equations (5)–(10) using Bayesian rules to obtain the joint density function of the observation X and all the i.i.d. latent variables $\mathcal{Z} = \{\mathbf{A}, \mathbf{B}, \mathbf{P}, \mathbf{Z}\}$. We assume that X is conditionally independent of P, given $\mathbf{Z}$. The joint density function is given by:

$$\begin{aligned} & f(\mathbf{X}, \mathcal{Z}) = f(\mathbf{X}, \mathbf{A}, \mathbf{B}, \mathbf{P}, \mathbf{Z}) \\ =& f(\mathbf{X}|\mathbf{A}, \mathbf{B}, \mathbf{Z}) f(\mathbf{Z}|\mathbf{P}) f(\mathbf{P}) f(\mathbf{A}) f(\mathbf{B}) \\ =& \prod_{n=1}^{N} \prod_{m=1}^{M} [p_m \mathbf{Beta}(\mathbf{x}_n|\mathbf{a}_m, \mathbf{b}_m)]^{z_{nm}} \cdot C(\mathbf{c}) \prod_{m=1}^{M} p_m^{c_m - 1} \\ & \cdot \prod_{l=1}^{L} \prod_{m=1}^{M} \left[\frac{\alpha_{lm}^{\mu_{lm}}}{\Gamma(\mu_{lm})} a_{lm}^{\mu_{lm}-1} e^{-\alpha_{lm} a_{lm}} \right] \cdot \left[\frac{\beta_{lm}^{\nu_{lm}}}{\Gamma(\nu_{lm})} b_{lm}^{\nu_{lm}-1} e^{-\beta_{lm} b_{lm}} \right] \end{aligned} \tag{A1}$$

and the logarithm of Equation (A1) is

$$\begin{aligned} & \mathcal{L}(\mathbf{X}, \mathcal{Z}) = \ln f(\mathbf{X}, \mathbf{Z}, \mathbf{A}, \mathbf{B}, \mathbf{P}) \\ =& \text{con.} + \sum_{n=1}^{N} \sum_{m=1}^{M} z_{nm} \left\{ \ln p_m + \sum_{l=1}^{L} \ln \frac{\Gamma(a_{lm} + b_{lm})}{\Gamma(a_{lm}) \Gamma(b_{lm})} \right. \\ & \left. + \sum_{l=1}^{L} [(a_{lm} - 1) \ln x_{ln} + (b_{lm} - 1) \ln(1 - x_{ln})] \right\} \\ & + \sum_{l=1}^{L} \sum_{m=1}^{M} [(\mu_{lm} - 1) \ln a_{lm} - \alpha_{lm} a_{lm}] \\ & + \sum_{l=1}^{L} \sum_{m=1}^{M} [(\nu_{lm} - 1) \ln b_{lm} - \beta_{lm} b_{lm}] + \sum_{m=1}^{M} (c_m - 1) \ln p_m \end{aligned} \tag{A2}$$

The i.i.d. latent variables we have now are $\mathbf{A}$, $\mathbf{B}$, $\mathbf{P}$ and $\mathbf{Z}$ with the hyperparameters $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, $\boldsymbol{\mu}$, $\boldsymbol{\nu}$ and $\mathbf{c}$. The updating scheme of the VI can be used to estimate these hyperparameters of the latent variables. Following the principles of the VI framework [15,32–35], we have:

$$c_m^* = c_{m_0} + \sum_{n=1}^{N} \mathbf{E}[z_{nm}] = c_{m_0} + \sum_{n=1}^{N} r_{nm} \tag{A3}$$

where c_{m_0} denotes the initial value of c_m.

We can also obtain the closed-form updating scheme for the hyperparameters α_{lm} and μ_{lm}:

$$\begin{aligned} \mu_{lm}^* =& \mu_{lm_0} + \sum_{n=1}^{N} \mathbf{E}[z_{nm}] \, \bar{a}_{lm} \left\{ \psi(\bar{a}_{lm} + \bar{b}_{lm}) - \psi(\bar{a}_{lm}) \right. \\ & \left. + \bar{b}_{lm} \cdot \psi'(\bar{a}_{lm} + \bar{b}_{lm}) (\mathbf{E}_b[\ln b_{lm}] - \ln \bar{b}_{lm}) \right\} \end{aligned} \tag{A4}$$

$$\alpha_{lm}^* = \alpha_{lm_0} - \sum_{n=1}^{N} \mathbf{E}[z_{nm}] \ln x_{ln} \tag{A5}$$

For the same reasons, the update schemes for the hyperparameters β_{lm} and ν_{lm} are:

$$\begin{aligned} \nu_{lm}^* =& \nu_{lm_0} + \sum_{n=1}^{N} \mathbf{E}[z_{nm}] \, \bar{b}_{lm} \left\{ \psi(\bar{a}_{lm} + \bar{b}_{lm}) - \psi(\bar{b}_{lm}) \right. \\ & \left. + \bar{a}_{lm} \cdot \psi'(\bar{a}_{lm} + \bar{b}_{lm}) (\mathbf{E}_a[\ln a_{lm}] - \ln \bar{a}_{lm}) \right\} \end{aligned} \tag{A6}$$

$$\beta_{lm}^* = \beta_{lm_0} - \sum_{n=1}^{N} \mathbf{E}[z_{nm}] \ln(1 - x_{ln}) \tag{A7}$$

Further details of the derivations are given in [26,27].

The update Equations (A3)–(A7) are calculated through the following expectations:

$$\begin{aligned}
&\overline{a} = \frac{\mu}{\alpha},\ \overline{b} = \frac{\nu}{\beta},\ \mathbf{E}\left[z_{nm}\right] = r_{nm},\ \mathbf{E}\left[\ln p_m\right] = \psi(c_m) - \psi(\widehat{c}),\\
&\mathbf{E}_a[\ln a] = \psi(\mu) - \ln\alpha,\ \mathbf{E}_b[\ln b] = \psi(\nu) - \ln\beta,\\
&\mathbf{E}_a\left[(\ln a - \ln\overline{a})^2\right] = \left[\psi(\mu) - (\ln\mu)\right]^2 + \psi^{'}(\mu)\\
&\mathbf{E}_b\left[(\ln b - \ln\overline{b})^2\right] = \left[\psi(\nu) - (\ln\nu)\right]^2 + \psi^{'}(\nu)
\end{aligned} \tag{A8}$$

B. *KS* and *L2* Statistics

To measure the closeness of fit between the generated graphs and the input graph, we use a Kolmogorov-Smirnov (*KS*) test and the *L2* (Euclidean) distance between the CCDFs for each of the statistics presented in Section 4. The model that generates graphs with the lowest *KS* and *L2* values has the closest fit to the real-world graph for that graph property.

The *KS* test is commonly used for goodness of fit and is based on the following statistic:

$$KS = \sup\left|F^*(x) - S(x)\right| \tag{B1}$$

where $F^*(x)$ is the hypothesised Cumulative Density Function (CDF) and $S(x)$ is the empirical distribution function based on the sampled data [36]. As trying to perform a linear fit on a heavy-tailed distribution is biased [36], we use the logarithmic variant of the *KS* test [14]:

$$KS = \sup\left|\log F^*(x) - \log S(x)\right| \tag{B2}$$

We also calculate the *L2* (Euclidean) distance for each statistic in the logarithmic scale:

$$L2 = \sqrt{\frac{1}{\log b - \log a}\sum_{x=a}^{b}(\log F^*(x) - \log S(x))^2} \tag{B3}$$

where $[a, b]$ is the support of distributions $F^*(x)$ and $S(x)$ respectively.

Note that for the Singular Values and Primary Left Singular Vectors, we use the standard (linear) version of *KS* and *L2*, as these statistics do not follow a heavy-tailed distribution.

B1. Real Attributes (Figures 5–8)

Table B1. *KS* Statistic for Undirected Graph, Real Attributes.

Statistic	Erdős-Rényi	MAG-R1		AGWAN-R1 (GMM)		AGWAN-R1 (BMM)	
		Age	Floor	Age	Floor	Age	Floor
Vertex Strength	6.064	5.940	5.799	2.303	2.303	1.609	2.303
Singular Values	3752.439	3809.375	3829.967	1284.796	507.622	263.853	262.647
Singular Vector	10.142	9.513	9.437	1.039	1.599	2.401	1.871
Clustering Coefficient	5.224	5.048	4.895	4.917	4.994	4.249	5.196
Triad Participation	7.012	6.877	6.685	6.646	6.646	6.898	6.685

Table B2. *L2* Statistic for Undirected Graph, Real Attributes.

Statistic	Erdős-Rényi	MAG-R1		AGWAN-R1 (GMM)		AGWAN-R1 (BMM)	
		Age	Floor	Age	Floor	Age	Floor
Vertex Strength	9.686	7.281	10.039	2.572	3.292	3.291	4.623
Singular Values	14,813.827	14,734.707	14,926.969	12,370.046	5624.668	2448.545	2772.618
Singular Vector	97.408	96.356	95.162	14.688	23.921	36.761	28.416
Clustering Coefficient	55.422	48.889	48.524	50.844	52.051	50.267	51.982
Triad Participation	16.879	17.334	17.101	17.419	19.601	19.596	19.726

Table B3. *KS* Statistic for Directed Graph, Real Attributes.

Statistic	Erdős-Rényi	MAG-R1	AGWAN-R1 (GMM)	AGWAN-R1 (BMM)
In-Vertex Strength	2.469	4.700	1.609	2.303
Out-Vertex Strength	2.708	2.659	1.204	2.303
Singular Values	40,564.590	7220.030	22,750.790	11,944.700
Singular Vector	8.927	9.154	0.714	0.619
Clustering Coefficient (In-Edges)	3.444	2.208	1.784	3.355
Clustering Coefficient (Out-Edges)	3.728	0.769	3.167	0.814
Clustering Coefficient	4.347	1.651	0.593	0.835
Triad Participation (Cycles)	4.787	4.248	3.219	4.248
Triad Participation (Middlemen)	4.382	4.500	2.639	2.503
Triad Participation (Ins)	4.700	4.500	3.401	4.248
Triad Participation (Outs)	4.382	4.094	1.851	4.248

Table B4. *L2* Statistic for Directed Graph, Real Attributes.

Statistic	Erdős-Rényi	MAG-R1	AGWAN-R1 (GMM)	AGWAN-R1 (BMM)
In-Vertex Strength	5.679	4.912	2.244	1.507
Out-Vertex Strength	5.100	3.534	2.463	2.510
Singular Values	262,699.994	32,990.793	104,552.681	60,843.087
Singular Vector	69.775	74.606	8.842	4.884
Clustering Coefficient (In-Edges)	3.528	1.607	1.215	2.231
Clustering Coefficient (Out-Edges)	3.145	1.191	1.891	1.399
Clustering Coefficient	6.949	1.438	0.813	1.033
Triad Participation (Cycles)	3.823	3.000	3.027	5.088
Triad Participation (Middlemen)	5.144	4.178	4.101	6.660
Triad Participation (Ins)	4.630	4.826	4.380	7.747
Triad Participation (Outs)	3.727	3.295	2.869	4.984

B2. Synthetic Attributes (Figures 9–12)

Table B5. *KS* Statistic for Undirected Graph, Synthetic Attributes.

Statistic	MAG			AGWAN (GMM)		
	L1	L5	L9	L1	L5	L9
Vertex Strength	2.243	5.670	6.234	2.207	1.354	0.722
Singular Values	6519.191	6580.987	8801.651	284.193	1518.976	4169.982
Singular Vector	5.037	3.447	0.362	3.514	2.686	2.082
Clustering Coefficient	1.283	4.401	4.773	4.627	4.422	2.319
Triad Participation	3.829	6.016	6.877	7.115	6.205	4.007

Statistic	AGWAN (BMM)				
	L1	L5	L9	R1 (Age)	R1 (Floor)
Vertex Strength	2.996	3.555	1.609	1.609	2.303
Singular Values	626.352	3206.064	4602.751	263.853	262.647
Singular Vector	3.212	2.334	2.408	2.401	1.871
Clustering Coefficient	4.784	4.960	3.257	4.249	5.196
Triad Participation	7.065	6.966	4.200	6.898	6.685

Table B6. *L2* Statistic for Undirected Graph, Synthetic Attributes.

Statistic	MAG			AGWAN (GMM)		
	L1	L5	L9	L1	L5	L9
Vertex Strength	7.944	10.103	21.027	4.980	3.611	2.570
Singular Values	32,272.685	32,524.858	47,155.221	2831.411	9885.424	30,238.426
Singular Vector	56.664	46.483	3.524	53.375	40.520	30.481
Clustering Coefficient	25.795	40.622	61.933	57.583	55.496	34.061
Triad Participation	12.047	11.038	29.136	22.053	20.686	9.375

Statistic	AGWAN (BMM)				
	L1	L5	L9	R1 (Age)	R1 (Floor)
Vertex Strength	5.610	8.036	3.812	3.291	4.623
Singular Values	6001.378	19,322.594	32,632.837	2448.545	2772.618
Singular Vector	49.353	35.109	35.913	36.761	28.416
Clustering Coefficient	57.318	54.340	36.950	50.267	51.982
Triad Participation	20.349	23.206	10.758	19.596	19.726

Table B7. *KS* Statistic for Directed Graph, Synthetic Attributes.

Statistic	MAG			AGWAN (GMM)		
	L1	L5	L9	L1	L5	L9
In-Vertex Strength	4.700	6.142	5.193	1.897	0.836	0.418
Out-Vertex Strength	4.942	6.234	3.401	1.157	0.691	2.303
Singular Values	7951.240	17,978.152	12,036.091	23,196.640	14,303.690	3181.910
Singular Vector	7.485	2.066	6.737	0.645	0.534	0.485
Clustering Coefficient (In-Edges)	2.961	4.578	4.512	3.298	2.535	2.370
Clustering Coefficient (Out-Edges)	3.164	5.463	2.865	4.830	3.140	1.717
Clustering Coefficient	3.278	5.839	4.000	4.996	2.949	2.558
Triad Participation (Cycles)	3.912	6.867	5.704	6.292	3.602	2.639
Triad Participation (Middlemen)	4.248	6.319	5.858	5.438	4.787	3.401
Triad Participation (Ins)	3.912	7.170	6.153	6.507	4.248	2.996
Triad Participation (Outs)	1.476	6.768	4.745	6.292	4.007	2.590

Statistic	AGWAN (BMM)			
	L1	L5	L9	R1
In-Vertex Strength	0.811	1.455	0.511	2.303
Out-Vertex Strength	2.148	0.693	2.303	2.303
Singular Values	15,219.730	12,480.590	1238.550	11,944.700
Singular Vector	0.775	0.451	0.451	0.619
Clustering Coefficient (In-Edges)	3.904	2.861	3.426	3.355
Clustering Coefficient (Out-Edges)	4.971	3.182	1.535	0.814
Clustering Coefficient	4.869	3.300	2.956	0.835
Triad Participation (Cycles)	6.380	5.561	3.912	4.248
Triad Participation (Middlemen)	6.620	5.768	3.401	2.503
Triad Participation (Ins)	6.579	5.075	2.526	4.248
Triad Participation (Outs)	6.492	5.394	2.303	4.248

Table B8. *L2* Statistic for Directed Graph, Synthetic Attributes.

Statistic	MAG			AGWAN (GMM)		
	L1	L5	L9	L1	L5	L9
In-Vertex Strength	5.023	15.718	7.066	2.109	2.189	0.649
Out-Vertex Strength	3.001	10.882	3.737	2.483	2.060	0.988
Singular Values	29,286.029	44,305.501	22,485.397	111,282.298	72,433.178	5554.271
Singular Vector	68.345	24.388	66.319	7.932	5.224	5.763
Clustering Coefficient (In-Edges)	2.507	7.692	4.819	3.731	1.710	1.594
Clustering Coefficient (Out-Edges)	2.771			5.642	2.785	1.501
Clustering Coefficient	2.450	13.922	7.653	8.296	2.617	1.827
Triad Participation (Cycles)	2.060	16.270	8.763	9.561	3.990	1.777
Triad Participation (Middlemen)	2.828	18.575	11.150	10.101	6.094	2.476
Triad Participation (Ins)	3.293	16.361	13.740	14.168	6.562	3.074
Triad Participation (Outs)	1.459	14.603	6.315	9.218	3.960	2.077

Statistic	AGWAN (BMM)			
	L1	L5	L9	R1
In-Vertex Strength	1.524	1.278	0.356	1.507
Out-Vertex Strength	2.731	1.524	0.907	2.510
Singular Values	78,886.496	54,111.356	2224.587	60,843.087
Singular Vector	9.494	5.531	5.153	4.884
Clustering Coefficient (In-Edges)	5.466	1.822	2.178	2.231
Clustering Coefficient (Out-Edges)	6.487	3.401	1.425	1.399
Clustering Coefficient	9.937	3.086	2.053	1.033
Triad Participation (Cycles)	11.684	7.300	3.946	5.088
Triad Participation (Middlemen)	13.301	11.218	3.997	6.660
Triad Participation (Ins)	14.616	11.028	3.766	7.747
Triad Participation (Outs)	9.419	8.347	2.527	4.984

References

1. Barabási, A.L.; Albert, R. Emergence of Scaling in Random Networks. *Science* **1999**, *286*, 509–512.
2. Chakrabarti, D.; Faloutsos, C. *Graph Mining: Laws, Tools, and Case Studies*; Morgan & Claypool Publishers: San Rafael, CA, USA, 2012.
3. Erdős, P.; Rényi, A. On the Evolution of Random Graphs. *Publ. Math. Inst. Hung. Acad. Sci.* **1960**, *5*, 17–61.
4. Leskovec, J.; Kleinberg, J.; Faloutsos, C. Graph evolution: Densification and shrinking diameters. *ACM Trans. Knowl. Discov. Data* **2007**, doi:10.1145/1217299.1217301.
5. Newman, M. *Networks: An Introduction*; OUP: New York, NY, USA, 2010.
6. Chakrabarti, D.; Zhan, Y.; Faloutsos, C. *R-MAT: A Recursive Model for Graph Mining*; In Proceedings of the 2004 SIAM International Conference on Data Mining, Lake Buena Vista, FL, USA, 2004, 442–446, doi:10.1137/1.9781611972740.43
7. Leskovec, J.; Chakrabarti, D.; Kleinberg, J.M.; Faloutsos, C.; Ghahramani, Z. Kronecker Graphs: An Approach to Modeling Networks. *J. Mach. Learn. Res.* **2010**, *11*, 985–1042.
8. Hoff, P.D.; Raftery, A.E.; Handcock, M.S. Latent Space Approaches to Social Network Analysis. *J. Am. Stat. Assoc.* **2002**, *97*, 1090–1098.
9. Kim, M.; Leskovec, J. Multiplicative Attribute Graph Model of Real-World Networks. *Internet Math.* **2012**, *8*, 113–160.
10. Wang, Y.; Wong, G. Stochastic Block Models for Directed Graphs. *J. Am. Stat. Assoc.* **1987**, *82*, 8–19.
11. Akoglu, L.; McGlohon, M.; Faloutsos, C. *OddBall*: *Spotting Anomalies in Weighted Graphs.*; Zaki, M.J., Yu, J.X., Ravindran, B., Pudi, V., Eds.; Lecture Notes in Computer Science; Springer, Berlin, Germany: 2010; Volume 6119, pp. 410–421.
12. Eichinger, F.; Huber, M.; Böhm, K. *On the Usefulness of Weight-Based Constraints in Frequent Subgraph Mining*; Bramer, M., Petridis, M., Hopgood, A., Eds.; Springer, Berlin, Germany: 2010; pp. 65–78.
13. Davis, M.; Liu, W.; Miller, P. Finding the most descriptive substructures in graphs with discrete and numeric labels. *J. Intell. Inf. Syst.* **2014**, *42*, 307–332.
14. Kim, M.; Leskovec, J. *Modeling Social Networks with Node Attributes Using the Multiplicative Attribute Graph Model*; Cozman, F.G., Pfeffer, A., Eds.; AUAI Press: Arlington, VA, USA: 2011; pp. 400–409.
15. Bishop, C.M. *Pattern Recognition and Machine Learning*, 3rd ed.; Springer: New York, NY, USA, 2011.
16. Jain, A.K.; Duin, R.P.W.; Mao, J. Statistical Pattern Recognition: A Review. *IEEE Trans. Pattern Anal. Mach. Intell.* **2000**, *22*, 4–37.
17. Figueiredo, M.A.T.; Jain, A.K. Unsupervised Learning of Finite Mixture Models. *IEEE Trans. Pattern Anal. Mach. Intell.* **2002**, *24*, 381–396.

18. Blei, D.M.; Jordan, M.I. Variational inference for Dirichlet process mixtures. *Bayesian Anal.* **2005**, *1*, 121–144.
19. Kurihara, K.; Welling, M.; Vlassis, N.A. *Accelerated Variational Dirichlet Process Mixtures*; Schölkopf, B., Platt, J.C., Hoffman, T., Eds.; MIT Press: Cambridge, MA, USA: 2006; pp. 761–768.
20. Davis, M.; Liu, W.; Miller, P. AGWAN: A Generative Model for Labelled, Weighted Graphs. In *New Frontiers in Mining Complex Patterns*, 2014; pp. 181–200.
21. Lindblom, J.; Samuelsson, J. Bounded support Gaussian mixture modeling of speech spectra. *IEEE Transa. Speech Audio Process.* **2003**, *11*, 88–99.
22. Bouguila, N.; Ziou, D.; Monga, E. Practical Bayesian estimation of a finite Beta mixture through Gibbs sampling and its applications. *Stat. Comput.* **2006**, *16*, 215–225.
23. Ji, Y.; Wu, C.; Liu, P.; Wang, J.; Coombes, K.R. Applications of beta-mixture models in bioinformatics. *Bioinformatics* **2005**, *21*, 2118–2122.
24. Ma, Z.; Leijon, A. Beta mixture models and the application to image classification. In Proceedings of 16th IEEE International Conference on Image Processing (ICIP), Cairo, Egypt, 7–10 November 2009; pp. 2045–2048.
25. McLachlan, G.J.; Krishnan, T. *The EM Algorithm and Extensions*, 1st ed.; Wiley: New York, NY, USA, 1997.
26. Ma, Z.; Leijon, A. Bayesian Estimation of Beta Mixture Models with Variational Inference. *IEEE Trans. Pattern Anal. Mach. Intell.* **2011**, *33*, 2160–2173.
27. Ma, Z.; Rana, P.K.; Taghia, J.; Flierl, M.; Leijon, A. Bayesian Estimation of Dirichlet Mixture Model with Variational Inference. *Pattern Recognit.* **2014**, *47*, 3143–3157.
28. Diaconis, P.; Ylvisaker, D. Conjugate Priors for Exponential Families. *Ann. Stat.* **1979**, *7*, 269–281.
29. Hunter, R.F.; Davis, M.; Tully, M.A.; Kee, F. *The Physical Activity Loyalty Card Scheme: Development and Application of a Novel System for Incentivizing Behaviour Change*; Kostkova, P., Szomszor, M., Fowler, D., Eds.; Springer: Berlin, Germany: 2011; Volume 91, pp. 170–177.
30. Fagiolo, G. Clustering in complex directed networks. *Phys. Rev. E* **2007**, doi:10.1103/PhysRevE.76.026107.
31. Johnson, D.B. Efficient Algorithms for Shortest Paths in Sparse Networks. *J. ACM* **1977**, *24*, 1–13.
32. Attias, H. *A Variational Bayesian Framework for Graphical Models*; Solla, S.A., Leen, T.K., Müller, K.R., Eds.; MIT Press: Cambridge, MA, USA, 1999; pp. 209–215.
33. Jaakkola, T.S. Tutorial on Variational Approximation Methods. In *Advanced Mean Field Methods: Theory and Practice*; Opper, M., Saad, D., Eds.; MIT Press: Cambridge, MA, USA, 2001; pp. 129–159.
34. Jaakkola, T.S.; Jordan, M.I. Bayesian parameter estimation via variational methods. *Stat. Comput.* **2000**, *10*, 25–37.

35. Ueda, N.; Ghahramani, Z. Bayesian model search for mixture models based on optimizing variational bounds. *Neural Netw.* **2002**, *15*, 1223–1241.
36. Goldstein, M.; Morris, S.; Yen, G. Problems with fitting to the power-law distribution. *Eur. Phys. J. B Condens. Matter Complex Syst.* **2004**, *41*, 255–258.