AN EFFICIENT BIPARTITE GRAPH SAMPLING ALGORITHM WITH PRESCRIBED DEGREE SEQUENCES

Tong Sun[1], Jianshu Hao[1], Zhiyang Zhang[2], and Guangxin Jiang[1]

[1]School of Management, Harbin Institute of Technology, Harbin, Heilongjiang, CHINA
[2]School of Science, Harbin University of Science and Technology, Harbin, Heilongjiang, CHINA

ABSTRACT

The structure of financial networks plays a crucial role in managing financial risks, particularly in the assessment of systemic risk. However, the true structure of these networks is often difficult to observe directly. This makes it essential to develop methods for sampling possible network configurations based on partial information, such as node degree sequences. In this paper, we consider the problem of sampling bipartite graphs (e.g., bank-asset networks) under such partial information. We first derive exact bounds on the number of nodes that can be connected at each step, given a prescribed degree sequence. Building on these bounds, we then introduce a weighted-balanced random sampling algorithm for generating bipartite graphs that are consistent with the observed degrees, and illustrate how the algorithm works through an example. In addition, we demonstrate the effectiveness of the proposed algorithm through numerical experiments.

1 INTRODUCTION

In finance, financial risk—especially systemic risk—poses a serious threat to market stability, presenting major challenges for regulators and policymakers. Systemic risk refers to the potential failure of one or more key financial institutions can spread through the financial network and potentially lead to a widespread collapse that affects the broader economy (Acharya et al. 2016). The structure of financial networks is a critical factor influencing the propagation of risk and, consequently, systemic risk (Ando et al. 2022). However, in practice, obtaining the true structure of financial networks is often difficult. Typically, only partial information is available, from which the possible network structure could be inferred. For example, we may know how many counterparties a financial institution interacts with or how many financial assets it holds, but not the specific institutions or assets involved (Glasserman and de Larrea 2023). Therefore, developing methods to infer (or sample) possible network structures from partial observations is an important problem.

In this paper, we focus on the bank-asset (or bank-firm loan) network, which captures the relationships between banks and various types of assets (loans) and is considered one of the most crucial networks within the financial system (Lux 2016). In such a network, a decline in the value of a single asset may force the institution holding it to sell other assets, thereby driving down their prices and causing losses for other institutions that hold the same or correlated assets (Glasserman and de Larrea 2018). In particular, we consider sampling the structure of the bank-asset network under partial information, that is, when only the number of assets held by each bank and the number of banks holding each asset are known.

The bank-asset network can be modeled as a bipartite graph, in which banks and assets constitute two disjoint sets of nodes. The ownership relationships, that is, which bank holds which asset, are represented by the edges connecting the corresponding nodes (Huang et al. 2013; Caccioli et al. 2018). The number of assets held by each bank, or the number of banks holding each asset, is indicated by the degree of the relevant node. The distribution of assets across banks and the variety of assets held by banks are captured by the degree sequences of the two disjoint sets of nodes. Therefore, as mentioned above, the partial information includes the number of assets held by each bank and the number of banks holding each asset, which corresponds to the degree of each node in the bipartite graph, also known as the degree sequence of the bipartite graph. Our goal is to sample bipartite graphs that satisfy this given degree sequence.

Previous studies have proposed a variety of methods to address the challenge of network reconstruction under partial information. One of the earliest and most intuitive approaches is the pairing model, also known as the configuration model (Wormald 1999), which constructs networks by randomly pairing node stubs according to a given degree sequence. However, this method often yields graphs containing multiple edges or self-loops—structural artifacts that are incompatible with the fundamental properties of bipartite graphs. Beyond this approach, three major classes of methods have been developed: Markov chain (MC) methods, swap-based methods, and maximum entropy (ME) methods. For MC methods, Diaconis and Sturmfels (1998) examined the use of Markov chains to sample matrices with specified marginals, proposing algorithms for both large-scale sampling and small-scale enumeration. Subsequent work extended these ideas through Markov chain Monte Carlo (MCMC) techniques for matrix sampling (Verhelst 2008; Gandy and Veraart 2017; Fosdick et al. 2018). These approaches typically begin with an initial network and define a Markov chain where each feasible network configuration represents a state, and transitions are governed by predefined probabilities. Swap-based methods start with a known network and iteratively select two independent edges for rewiring. For example, edges (a,b) and (c,d) can be replaced with (a,c) and (b,d). This procedure, often referred to as the switching method (Carstens and Horadam 2016; Wormald 1999), preserves the degree sequences while exploring the space of feasible networks. In the ME framework, Barvinok (2010) established a foundational connection between random binary matrices with fixed row and column sums and a corresponding maximum entropy matrix. Building on this, Glasserman and de Larrea (2023) investigated ME-based approaches for graph simulation and proposed a sequential sampling algorithm for generating bipartite graphs consistent with prescribed degree sequences.

Despite their methodological differences, each of these three classes of approaches has notable limitations. Both MC and swap-based methods struggle to ensure independence between generated samples and depend heavily on well-initialized networks. MC methods, in particular, require a "burn-in" period to reach stationarity, while swap-based methods often necessitate tens of thousands of edge swaps—commonly over 50,000—to achieve adequate randomization (Fayle and Manica 2010). Although ME methods avoid the issue of sample dependence, they involve repeatedly solving complex optimization problems, leading to significant computational overhead and limited scalability for large networks. These challenges underscore the need for more efficient methods capable of directly generating independent samples from given degree sequences without sacrificing computational performance.

To address these limitations, we propose an efficient algorithm for rapidly and randomly sampling bipartite graphs consistent with prescribed degree sequences. The algorithm begins by sorting the two degree sequences in non-increasing and non-decreasing order, respectively. Nodes in the non-decreasing sequence are then grouped by degree. At each step, the algorithm connects the highest-degree node from the non-increasing sequence to nodes in these groups, according to a connection rule derived from exact bounds on the number of nodes that can be connected. This process is iterated until all connections are formed and the bipartite graph is complete. A formal proof of the theorem, which establishes the exact bounds on the number of nodes that can be connected at each step, will be provided in the full journal version of this work.

The remainder of the paper is organized as follows. Section 2 formulates the problem of bipartite graph sampling with prescribed degree sequences. Section 3 presents the theorem underpinning the proposed algorithm, provides the complete pseudocode, and illustrates the algorithm's implementation with a simple example. In Section 4, we evaluate the performance of the proposed algorithm through comprehensive numerical experiments, using the sequential algorithm from Glasserman and de Larrea (2023) as a benchmark. Section 5 concludes this paper.

2 PROBLEM FORMULATION

Consider a financial network comprising m assets and n banks, where each bank holds a subset of the available assets. This bank-asset network can be modeled as a bipartite graph, in which one set of nodes corresponds to assets, the other to banks, and each edge indicates that a given bank holds a particular asset. In such a network, the degree of an asset node represents the number of banks that hold the asset, while the degree of a bank node indicates the number of distinct assets owned by that bank. Given the degree sequences of both asset and bank nodes, our goal is to sample bipartite graphs that are consistent with this partial information.

Let $\mathbf{G}(\mathbf{a},\mathbf{b},\mathbf{E})$ denote the set of all bipartite graphs with disjoint vertex sets whose degree sequences corresponding to the given positive integer vectors $\mathbf{a} \in \mathbb{N}_{+}^{m}$ and $\mathbf{b} \in \mathbb{N}_{+}^{n}$, where $\mathbf{E}$ denotes the edge set. Let $G(\mathbf{a},\mathbf{b},\mathbf{E})$ be an arbitrary element (a bipartite graph) of $\mathbf{G}(\mathbf{a},\mathbf{b},\mathbf{E})$. Without loss of generality, we assume that the asset degree sequence $\mathbf{a}$ is sorted in non-decreasing order

$$0 < [\mathbf{a}]_1 \le [\mathbf{a}]_2 \le \cdots \le [\mathbf{a}]_m,$$

where $[\mathbf{x}]_i$ denotes the i-th entry of vector $\mathbf{x}$, and the bank degree sequence $\mathbf{b}$ is sorted in non-increasing order

$$[\mathbf{b}]_1 \ge [\mathbf{b}]_2 \ge \cdots \ge [\mathbf{b}]_n > 0.$$

Example 1 We consider the degree sequences $\mathbf{a} = (1,1,2)$ and $\mathbf{b} = (2,1,1)$, i.e., there are three assets and three banks. Two of the assets are each held by one bank, and one asset is held by two banks. Similarly, two of the banks each hold one asset, and one bank holds two assets. By enumeration, we could find that there are five bipartite graphs consistent with this degree sequences, as shown in Figure 1. The goal of this paper is to develop an efficient algorithm for sampling bipartite graphs from the space $\mathbf{G}(\mathbf{a},\mathbf{b},\mathbf{E})$, such as the five bipartite graphs shown, while ensuring that each sampled graph satisfies the prescribed degree sequences.

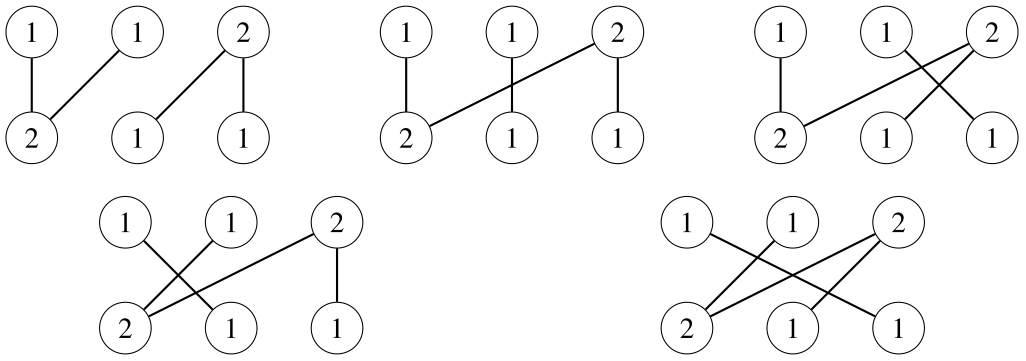

Figure 1: All five bipartite graphs consistent with $\mathbf{a} = (1,1,2)$ and $\mathbf{b} = (2,1,1)$.

3 EFFICIENT BIPARTITE GRAPH SAMPLING ALGORITHM

In this section, we present an efficient algorithm for sampling bipartite graphs with prescribed degree sequences. We begin by introducing the main mechanism of the proposed algorithm. Section 3.1 states the theorem on which the algorithm is based. Section 3.2 provides a detailed explanation of the algorithm, including the complete pseudocode. Section 3.3 illustrates the implementation of the algorithm through a simple example.

We begin by describing the fundamental idea behind the proposed algorithm. First, the asset nodes in $\mathbf{a}$ are partitioned into p groups based on distinct degree values. Specifically, group k contains m_k nodes of degree α_k, where $\alpha_1 < \alpha_2 < \cdots < \alpha_p$ and $\sum_{k=1}^{p} m_k = m$. Next, the bank node $[\mathbf{b}]_1$ selects the number of asset nodes to connect to, proceeding sequentially from group 1 to group p. After completing its connections, any nodes whose degrees have been reduced to zero are removed. The updated degree sequences are denoted as $\mathbf{a}^1 \in \mathbb{N}_+^{m^1}$ and $\mathbf{b}^1 \in \mathbb{N}_+^{n-1}$. Here, m^1 denotes the number of remaining asset nodes which depends on the number of nodes removed, and the number of bank nodes reduces to $n-1$. These grouping and connection steps are then applied iteratively until all banks nodes have been processed. Importantly, each iteration can be regarded as solving a new instance of the same problem with updated prescribed degree sequences. Therefore, it is sufficient to focus on the selection process for the first bank node $[\mathbf{b}]_1$.

Return to Example 1 with $\mathbf{a} = (1,1,2)$ and $\mathbf{b} = (2,1,1)$, we use it to illustrate the fundamental idea behind our algorithm. First, the asset nodes in $\mathbf{a}$ are partitioned into two groups ($p = 2$): group 1 contains 2 nodes with degree 1 ($\alpha_1 = 1$, $m_1 = 2$), and group 2 contains 1 node with degree 2 ($\alpha_2 = 2$, $m_2 = 1$). The bank node, $[\mathbf{b}]_1$, with degree 2, then sequentially selects asset nodes, starting from group 1 and proceeding to group 2. For instance, $[\mathbf{b}]_1$ connects to two nodes from group 1, reducing their degree from 1 to 0. After completing the connections, the two nodes with degree 0 in group 1 are removed, resulting in the updated degree sequence $\mathbf{a}^1 = (2)$ for the remaining asset node, and $\mathbf{b}^1 = (1,1)$ for the bank nodes. This process repeats for the next bank node using the updated degree sequences, continuing until all banks nodes have been processed.

3.1 Node Selection Bound

In this subsection, we provide the exact bounds on the number of nodes involved in the sequential selection process. Before deriving the exact bounds, we introduce the following assumption regarding the existence of a bipartite graph with the prescribed degree sequences, which can be readily verified using the Gale-Ryser theorem (see, for instance, Gale 1957).

Assumption 1 A bipartite graph exists for the prescribed degree sequences $\mathbf{a}$ and $\mathbf{b}$.

We then introduce several essential mathematical definitions required for the formulation of the exact bounds. Define $\mathscr{L} : (\mathbb{N}^p, \mathbb{N}_+) \to \{0,1\}^{p\times c}$ as a matrix operation that maps a vector $\mathbf{x} \in \mathbb{N}^p$ and a positive integer $c \in \mathbb{N}_+$ to a binary matrix of size $p \times c$. The operation is defined element-wise as follows:

$$[\mathscr{L}(\mathbf{x},c)]_{ij} = \begin{cases} 1, & \text{if } 1 \le j \le [\mathbf{x}]_i; \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

Here, the resulting binary matrix $\mathscr{L}(\mathbf{x},c)$ is such that for each $i \in \{1,2,\ldots,p\}$, the first $\mathbf{x}_i$ columns in row i are set to 1, while the remaining entries are set to 0.

Let $\mathbf{z} \in \mathbb{N}^{|\mathbf{b}|}$ denote a vector derived from the matrix $\mathscr{L}(\mathbf{a},|\mathbf{b}|)$, defined as follows:

$$[\mathbf{z}]_j = \sum_{i=1}^{m} [\mathscr{L}(\mathbf{a},|\mathbf{b}|)]_{ij}, \tag{2}$$

where $[\mathbf{z}]_j$ represents the sum of the j-th column of $\mathscr{L}(\mathbf{a},|\mathbf{b}|)$.

To illustrate the above definitions, we return to the earlier example with $\mathbf{a} = (1,1,2)$ and $\mathbf{b} = (2,1,1)$. By (1) and (2), we obtain:

$$\mathscr{L}(\mathbf{a},|\mathbf{b}|) = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \end{bmatrix},$$

and $\mathbf{z} = (3,1,0)$.

Define $\mathscr{D} : (\mathbb{N}^p, \mathbb{N}^q) \to \mathbb{Z}^{\max\{p,q\}}$ as a vector operation that maps two vectors $\mathbf{x} \in \mathbb{N}^p$ and $\mathbf{y} \in \mathbb{N}^q$ to another vector of length $\max\{p,q\}$. The operation is defined element-wise as follows:

$$[\mathscr{D}(\mathbf{x},\mathbf{y})]_h = \begin{cases} [\mathbf{x}]_h - [\mathbf{y}]_h, & \text{if } h \le \min\{p,q\}; \\ [\mathbf{x}]_h, & \text{if } h > \min\{p,q\} \text{ and } p \ge q; \\ -[\mathbf{y}]_h, & \text{if } h > \min\{p,q\} \text{ and } p < q. \end{cases} \tag{3}$$

Here, the resulting vector $\mathscr{D}(\mathbf{x},\mathbf{y})$ is such that for each $h \in \{1,2,\dots,\max\{p,q\}\}$, the value is given by $[\mathbf{x}]_h - [\mathbf{y}]_h$, with zeros appended to either $\mathbf{x}$ or $\mathbf{y}$ if their lengths are less than $\max\{p,q\}$.

Define $\mathscr{F} : \mathbb{N}_+ \to \mathbb{N}$ as an operation that maps a positive integer $c \in \mathbb{N}_+$ to a natural number. Let $\mathscr{F}(k) \in [0, m_k] \cap \mathbb{N}$ (for $k = 1,2,\dots,p$) denote the number of nodes selected by $[\mathbf{b}]_1$ from the m_k nodes in group k of $\mathbf{a}$. After selecting the number of nodes in group $k-1$, the following definitions are introduced:

- The updated $\mathbf{a}$, denoted by $\mathbf{a}^{(k)}$, where $\mathbf{a}^{(1)} = \mathbf{a}$;
- The updated $\mathbf{z}$ derived from $\mathscr{Z}\left(\mathbf{a}^{(k)}, |\mathbf{b}|\right)$, denoted by $\mathbf{z}^{(k)}$, where $\mathbf{z}^{(1)} = \mathbf{z}$;
- The remaining degree of $[\mathbf{b}]_1$, denoted by $[\mathbf{b}]_1^{(k)}$, where $[\mathbf{b}]_1^{(k)} = [\mathbf{b}]_1 - \sum_{k'=1}^{k-1} \mathscr{F}(k')$ and $[\mathbf{b}]_1^{(1)} = [\mathbf{b}]_1$;
- The total number of nodes in groups $k+1$ to p, denoted by $M^{(k)}$, where $M^{(k)} = \sum_{k'=k+1}^{p} m_{k'}$ and $M^{(p)} = 0$.

By combining $\mathbf{z}^{(k)}$, $\mathbf{b}^1$, and (3), we introduce the following definitions that will be used in deriving the exact bounds:

$$\mathscr{D}_{left}^{(k)} = \sum_{h=1}^{\alpha_k - 1} \left[\mathscr{D}\left(\mathbf{z}^{(k)}, \mathbf{b}^1\right)\right]_h, \quad \mathscr{D}_{now}^{(k)} = \left[\mathscr{D}\left(\mathbf{z}^{(k)}, \mathbf{b}^1\right)\right]_{\alpha_k},$$

and

$$\mathscr{D}_{right}^{(k)} = \min\left\{ \min_r \sum_{h=\alpha_k+1}^{r} \left[\mathscr{D}\left(\mathbf{z}^{(k)}, \mathbf{b}^1\right)\right]_h, 0 \right\},$$

where $\mathscr{D}\left(\mathbf{z}^{(k)}, \mathbf{b}^1\right) \in \mathbb{Z}^n$. Notice that the vector $\mathbf{z}^{(k)}$, derived from the matrix $\mathscr{Z}\left(\mathbf{a}^{(k)}, |\mathbf{b}|\right)$, and the vector $\mathbf{b}^1$ are not in the same state. Nevertheless, we apply the $\mathscr{D}$-operation to both vectors for the theorem.

In this paper, we adopt the following notational conventions for summation operator: The summation operator $\sum$ is defined to be zero whenever its upper limit is less than its lower limit. Additionally, any term whose index falls outside the summation range is treated as zero.

Based on the above mathematical definitions, we establish the following theorem, which provides the exact bounds for $\mathscr{F}(k)$ (for $k = 1,2,\dots,p$). Here, $\mathscr{F}(k)$ denotes the number of nodes selected by $[\mathbf{b}]_1$ from the m_k nodes in group k of $\mathbf{a}$.

Theorem 1 Under Assumption 1, the number of nodes $\mathscr{F}(k)$ selected in group k of $\mathbf{a}$, has the lower bound $\max\left\{[\mathbf{b}]_1^{(k)} - M^{(k)}, 0\right\}$ and the upper bound $\min\left\{\mathscr{D}_{\text{left}}^{(k)} + \mathscr{D}_{\text{now}}^{(k)} + \mathscr{D}_{\text{right}}^{(k)}, m_k\right\}$, and can take any integer value within this range, that is,

$$\mathscr{F}(k) \in \left[\max\left\{[\mathbf{b}]_1^{(k)} - M^{(k)}, 0\right\},\ \min\left\{\mathscr{D}_{\text{left}}^{(k)} + \mathscr{D}_{\text{now}}^{(k)} + \mathscr{D}_{\text{right}}^{(k)}, m_k\right\}\right] \cap \mathbb{N}.$$

3.2 Sampling Algorithm

Given degree sequences $\mathbf{a}$ and $\mathbf{b}$ that satisfy Assumption 1, our goal is to sample random bipartite graphs consistent with these sequences. According to Theorem 1, for each bank node $[\mathbf{b}]_j$, we can determine the number $\mathscr{F}(k)$ of asset nodes to connect to from each group in $\mathbf{a}$. To achieve uniform sampling over the space of all valid bipartite graphs, we randomly select $\mathscr{F}(k)$ within its exact bounds, assigning a weight to each possible value proportional to the number of distinct bipartite graphs it can produce.

Specifically, the weight of $\mathscr{F}(k)$ is determined by the following three processes:

1. Select $\mathscr{F}(k)$ nodes from group k in $\mathbf{a}$. The number of bipartite graphs generated in this step is denoted by N_1, given by
$$N_1 = \binom{m_k}{\mathscr{F}(k)}.$$
2. Select $\mathscr{F}(k+1), \ldots, \mathscr{F}(p)$ nodes from groups $k+1, \ldots, p$ in $\mathbf{a}$, respectively. The number of bipartite graphs generated here is denoted by N_2, which we approximate as
$$\hat{N}_2 = \binom{M^{(k)} + m_k - \mathscr{F}(k)}{[\mathbf{b}]_j^k - \mathscr{F}(k)}.$$
3. Generate the bipartite graph with update $\mathbf{a}$ and $\mathbf{b}$ after processing $[\mathbf{b}]_j$. For computational efficiency, we assume that the number of bipartite graphs generated in this step remains the same for different values of $\mathscr{F}(k)$.

Therefore, the weight of $\mathscr{F}(k)$ is $N_1 \times \hat{N}_2$, which forms the normalized weight vector $\hat{\mathbf{w}}$ for sampling $\mathscr{F}(k)$ within its exact bounds. Notably, further refinement of $\hat{\mathbf{w}}$ could enhance the uniformity of the bipartite graph sampling distribution, which is part of our ongoing work.

By leveraging Theorem 1 and the weight vector $\hat{\mathbf{w}}$, we present the pseudocode of the proposed algorithm in Algorithm 1.

Algorithm 1 Efficient Bipartite Graph Sampling (EBGS) Algorithm.

```
Require: A degree sequence a in non-decreasing order, a degree sequence b in non-increasing order, and
    an initially empty bipartite graph $G(\mathbf{a},\mathbf{b},\mathbf{E})$ where $\mathbf{E} = \emptyset$.
Ensure: A bipartite graph $G(\mathbf{a},\mathbf{b},\mathbf{E})$.
 1: for $j = 1$ to $n$ do
 2:     Group the nodes in a: Partition the nodes into $p$ groups based on their degree values, where group
        $k$ contains $m_k$ nodes with degree $\alpha_k$, satisfying $\alpha_1 < \alpha_2 < \cdots < \alpha_p$ and $\sum_{k=1}^{p} m_k = m$.
 3:     for $k = 1$ to $p$ do
 4:         Compute the following quantities:
                $[\mathbf{b}]_j^{(k)}, \quad M^{(k)}, \quad \mathscr{D}_{\text{left}}^{(k)}, \quad \mathscr{D}_{\text{now}}^{(k)}, \quad \mathscr{D}_{\text{right}}^{(k)}, \quad \hat{\mathbf{w}}.$
 5:         Sample an integer $\mathscr{F}(k)$: Select $\mathscr{F}(k)$ randomly from
                $\left[\max\left\{[\mathbf{b}]_j^{(k)} - M^{(k)}, 0\right\},\ \min\left\{\mathscr{D}_{\text{left}}^{(k)} + \mathscr{D}_{\text{now}}^{(k)} + \mathscr{D}_{\text{right}}^{(k)}, m_k\right\}\right]$
            according to the weight vector $\hat{\mathbf{w}}$.
 6:         Random selection of nodes to connect: Randomly choose $\mathscr{F}(k)$ nodes of degree $\alpha_k$ from
            group $k$ to connect with the node $[\mathbf{b}]_j$, and add the corresponding edges to E.
 7:     end for
 8:     Update a based on the newly added edges in E.
 9: end for
10: Output: $G(\mathbf{a},\mathbf{b},\mathbf{E})$.
```

The algorithm starts with two degree sequences, $\mathbf{a}$ and $\mathbf{b}$, sorted in non-decreasing and non-increasing order, respectively, along with an initially empty bipartite graph $G(\mathbf{a},\mathbf{b},\mathbf{E})$. For each node $[\mathbf{b}]_j$, the nodes in $\mathbf{a}$ are partitioned into groups based on their distinct degree values α_k, where each group k contains m_k nodes and $\sum_{k=1}^{p} m_k = m$. For each group k, we compute a set of auxiliary parameters that specify an exact bound for $\mathscr{F}(k)$, as well as a weight vector $\hat{\mathbf{w}}$ used to sample $\mathscr{F}(k)$ within this bound. Once $\mathscr{F}(k)$ has been determined, we randomly select $\mathscr{F}(k)$ nodes from group k to connect to the current node $[\mathbf{b}]_j$, and add the corresponding edges to $\mathbf{E}$. After processing $[\mathbf{b}]_j$, we update $\mathbf{a}$ by decrementing the degrees of the newly connected nodes, ensuring that the degree sequence remains accurate. Finally, after processing all nodes in $\mathbf{b}$, the algorithm generates a bipartite graph $G(\mathbf{a},\mathbf{b},\mathbf{E})$ that satisfies the prescribed degree sequences.

3.3 Algorithm Illustration

To illustrate our algorithm more clearly, we revisit Example 1. Given the degree sequences $\mathbf{a} = (1,1,2)$ and $\mathbf{b} = (2,1,1)$, it is straightforward to determine that $|\mathbf{G}(\mathbf{a},\mathbf{b},\mathbf{E})| = 5$. Figure 2 shows one realization of Algorithm 1.

(1) For $[\mathbf{b}]_1$:

$\mathscr{F}(1) \in \{1,2\}$ $\quad \hat{\mathbf{w}} = (0.8, 0.2)$ $\quad \mathscr{F}(1) = 1$

$\mathscr{F}(2) \in \{1\}$ $\quad \hat{\mathbf{w}} = (1)$ $\quad \mathscr{F}(2) = 1$

(2) For $[\mathbf{b}]_2$:

$\mathscr{F}(1) \in \{1\}$ $\quad \hat{\mathbf{w}} = (1)$ $\quad \mathscr{F}(1) = 1$

(3) For $[\mathbf{b}]_3$:

$\mathscr{F}(1) \in \{1\}$ $\quad \hat{\mathbf{w}} = (1)$ $\quad \mathscr{F}(1) = 1$

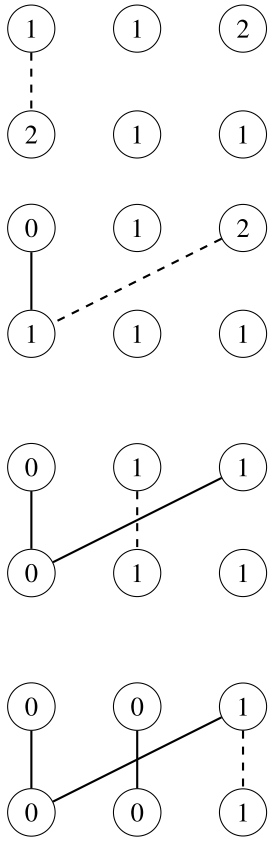

Figure 2: An illustration of Algorithm 1 with $\mathbf{a} = (1,1,2)$ and $\mathbf{b} = (2,1,1)$.

In Step 1 for $[\mathbf{b}]_1$, we start by partitioning $\mathbf{a}$ into two groups based on their degree values. For group 1, we compute the auxiliary parameters ($[\mathbf{b}]_1^{(1)} = 2,\ M^{(1)} = 1,\ \mathscr{D}_{\text{left}}^{(1)} = 0,\ \mathscr{D}_{\text{now}}^{(1)} = 2,\ \mathscr{D}_{\text{right}}^{(1)} = 0,\ m_1 = 2$), from which the number of nodes to connect is determined via Theorem 1 as

$$\mathscr{F}(1) \in \left[\max\left\{[\mathbf{b}]_1^{(1)} - M^{(1)}, 0\right\},\ \min\left\{\mathscr{D}_{\text{left}}^{(1)} + \mathscr{D}_{\text{now}}^{(1)} + \mathscr{D}_{\text{right}}^{(1)}, m_1\right\}\right] \cap \mathbb{N} = \{1,2\},$$

with an associated weight vector $\hat{\mathbf{w}} = (0.8, 0.2)$. This indicates that $\mathscr{F}(1) = 1$ is selected with probability 0.8, and $\mathscr{F}(1) = 2$ with probability 0.2, and we suppose that the random selection yields $\mathscr{F}(1) = 1$. We then randomly select one node from group 1 to connect to $[\mathbf{b}]_1$. For group 2, we repeat the same procedure, computing the auxiliary parameters ($[\mathbf{b}]_1^{(2)} = 1,\ M^{(2)} = 0,\ \mathscr{D}_{\text{left}}^{(2)} = 1,\ \mathscr{D}_{\text{now}}^{(2)} = 0,\ \mathscr{D}_{\text{right}}^{(2)} = 0,\ m_2 = 1$), from which we obtain

$$\mathscr{F}(2) \in \Big[\max\big\{[\mathbf{b}]_1^{(2)} - M^{(2)}, 0\big\},\ \min\big\{\mathscr{D}_{\text{left}}^{(2)} + \mathscr{D}_{\text{now}}^{(2)} + \mathscr{D}_{\text{right}}^{(2)}, m_2\big\}\Big] \cap \mathbb{N} = \{1\},$$

with weight vector $\hat{\mathbf{w}} = (1)$. This indicates that $\mathscr{F}(2) = 1$ is deterministically selected, and the sole node in group 2 is thus assigned to connect to $[\mathbf{b}]_1$. Finally, we update $\mathbf{a}$ by decrementing both $[\mathbf{a}]_1$ and $[\mathbf{a}]_3$ by 1 to reflect the new connections.

In Step 2 for $[\mathbf{b}]_2$, we again partition the updated $\mathbf{a}$ based on degree values. At this stage, only one group is present. We compute the auxiliary parameters ($[\mathbf{b}]_2^{(1)} = 1$, $M^{(1)} = 0$, $\mathscr{D}_{\text{left}}^{(1)} = 0$, $\mathscr{D}_{\text{now}}^{(1)} = 1$, $\mathscr{D}_{\text{right}}^{(1)} = 0$, $m_1 = 2$), leading to

$$\mathscr{F}(1) \in \Big[\max\big\{[\mathbf{b}]_2^{(1)} - M^{(1)}, 0\big\},\ \min\big\{\mathscr{D}_{\text{left}}^{(1)} + \mathscr{D}_{\text{now}}^{(1)} + \mathscr{D}_{\text{right}}^{(1)}, m_1\big\}\Big] \cap \mathbb{N} = \{1\},$$

with weight vector $\hat{\mathbf{w}} = (1)$. We then randomly select one node from this group to connect to $[\mathbf{b}]_2$. Finally, we update $\mathbf{a}$ by decrementing $[\mathbf{a}]_2$ by 1.

In Step 3 for $[\mathbf{b}]_3$, only one node with degree 1 remains in each of $\mathbf{a}$ and $\mathbf{b}$. These nodes are directly connected, which is consistent with the theoretical result that $\mathscr{F}(1) = 1$. Specifically, we compute the auxiliary parameters ($[\mathbf{b}]_3^{(1)} = 1$, $M^{(1)} = 0$, $\mathscr{D}_{\text{left}}^{(1)} = 0$, $\mathscr{D}_{\text{now}}^{(1)} = 1$, $\mathscr{D}_{\text{right}}^{(1)} = 0$, $m_1 = 1$), leading to

$$\mathscr{F}(1) \in \Big[\max\big\{[\mathbf{b}]_3^{(1)} - M^{(1)}, 0\big\},\ \min\big\{\mathscr{D}_{\text{left}}^{(1)} + \mathscr{D}_{\text{now}}^{(1)} + \mathscr{D}_{\text{right}}^{(1)}, m_1\big\}\Big] \cap \mathbb{N} = \{1\},$$

with weight vector $\hat{\mathbf{w}} = (1)$.

4 NUMERICAL EXPERIMENTS

In this section, we evaluate the performance of our EBGS algorithm (Algorithm 1) in terms of sampling speed and uniformity through a comparative analysis with the *sequential algorithm* presented in Glasserman and de Larrea (2023), which is regarded as one of the most effective methods for bipartite graph reconstruction with prescribed degree sequences. All numerical experiments are implemented in Python and executed on a desktop computer equipped with a 3.6 GHz Intel Core i7-12700K processor and 16 GB of RAM.

4.1 Sampling Speed

To ensure a fair comparison, we reproduce the sequential algorithm presented in Glasserman and de Larrea (2023) and apply it to the same set of degree sequences used for bipartite graph reconstruction in their experiments:

- Interbank-1: $\mathbf{a} = \mathbf{b} = (6,6,6,5,5,5,5,4,4,3,2)$.
- Interbank-2: $\mathbf{a} = \mathbf{b} = (9,9,9,9,9,9,8,8,8,7,6)$.
- Chesapeake:
  $\mathbf{a} = (7,8,5,1,1,1,5,7,1,0,1,2,0,5,6,2,0,6,2,0,1,6,3,0,0,0,1,0,0,0,1,0,0)$,
  $\mathbf{b} = (0,0,0,0,0,1,3,3,3,2,3,3,3,1,1,1,2,1,6,1,1,3,3,1,3,4,5,3,3,3,1,4,4)$.

We then implement Algorithm 1 using the same degree sequences and compare the average runtime required to sample a single random bipartite graph with that of the sequential algorithm. Both algorithms are used to generate 1,000 samples. Table 1 reports the results for the original implementation of the sequential algorithm, our reproduction, and our proposed algorithm. As shown in the table, our reproduction results closely aligns with the results in the original paper, confirming the correctness of our implementation. Moreover, our algorithm achieves a substantial speedup compared to the sequential algorithm, with improvements of more than two orders of magnitude across all tested degree sequences.

Table 1: Comparison of the average runtime between the EBGS algorithm and the sequential algorithm.

	Sequential Algorithm		EBGS Algorithm	Speedup Ratio
	Original	Reproduced		
Interbank-1	0.17s	0.20s	0.0014s	142.86
Interbank-2	0.15s	0.19s	0.0014s	135.71
Chesapeake	2.24s	2.69s	0.0043s	625.58

Furthermore, we compare the average runtime required by our algorithm and the sequential algorithm to sample a single random bipartite graph as the number of nodes increases. Specifically, we construct the following degree sequences, which ensure the existence of a corresponding bipartite graph as the number of nodes increases:

$$\mathbf{a} = \mathbf{b} = (n-1, n-1, n-2, n-3, \ldots, 1),$$

where n is the number of nodes. We implement our algorithm using the degree sequence with n ranging from 3 to 300, and the sequential algorithm with n ranging from 3 to 74 (the average runtime of the sequential algorithm exceeds 100 seconds when $n > 74$).

Figure 3 presents the comparison of the average runtime against the number of nodes for both our algorithm and the sequential algorithm, while Figure 4 reports the corresponding speedup achieved by our algorithm. As shown in the figures, our algorithm significantly outperforms the sequential algorithm, with the speedup increasing approximately linearly as n grows. For example, when $n = 24$, the sequential algorithm takes 1.29 seconds to sample a bipartite graph, whereas our algorithm requires only 0.012 seconds, achieving a speedup of over 100 times. When $n = 47$, the sequential algorithm takes 14.11 seconds, while our algorithm takes only 0.070 seconds, resulting in a speedup exceeding 200 times. Similarly, for $n = 70$, the sequential algorithm takes 75.36 seconds, whereas our algorithm requires only 0.25 seconds, achieving a speedup of over 300 times. If the budget of runtime is limited to 50 seconds, the sequential algorithm can only sample a bipartite graph for $n < 64$, while our algorithm can handle instances with $n > 300$. Specifically, our algorithm can sample a bipartite graph with $n = 52$ in 0.1 seconds, $n = 106$ in 1 second, and $n = 199$ in 10 seconds, demonstrating its high efficiency in bipartite graph sampling.

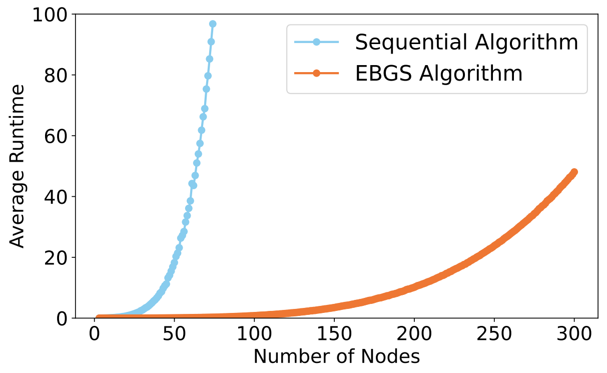

Figure 3: Comparison of the average runtime versus the number of nodes for both algorithms.

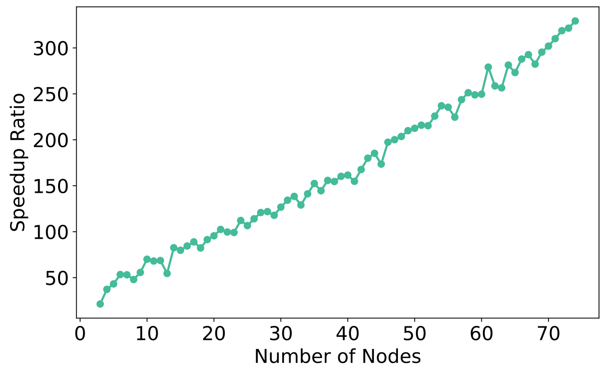

Figure 4: Speedup ratio of the EBGS algorithm relative to the sequential algorithm.

4.2 Sampling Uniformity

To evaluate sampling uniformity, we implement the EBGS algorithm and the sequential algorithm with the most uniform adaptive order rule on four randomly constructed degree sequences. For each degree sequence, both algorithms are used to generate 5,000 samples. The uniformity of the resulting sampling

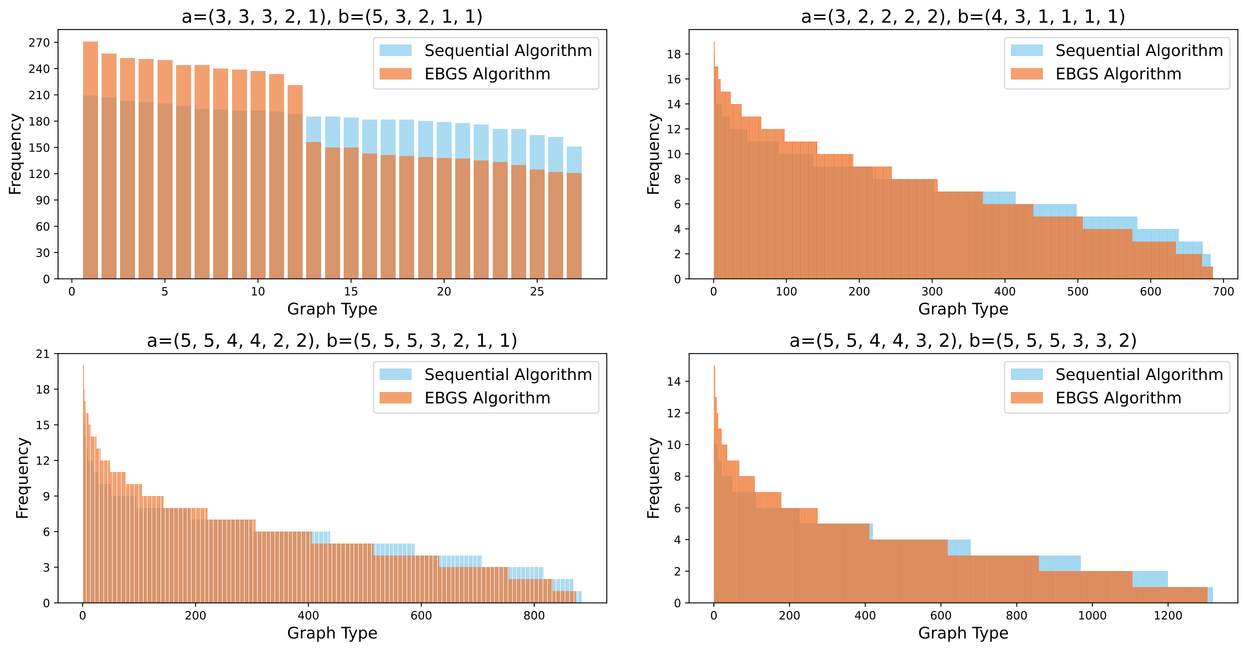

Figure 5: The sampling distributions of the EBGS algorithm and the sequential algorithm.

distributions is assessed using two metrics: the coefficient of variation (CV) of each sampling distribution, and the Kullback–Leibler (KL) divergence between the sampling distribution and the corresponding uniform distribution.

Figure 5 presents the detailed sampling distributions obtained under the four degree sequences, where the horizontal axis ranks bipartite graphs from most to least frequent and the vertical axis indicates their frequencies. Table 2 reports the number of distinct bipartite graphs generated (denoted as N), as well as the associated CV and KL values for both algorithms. The results show that, compared to the sequential algorithm, our algorithm generates a similar number of distinct bipartite graphs. However, it exhibits slightly higher CV and KL values, indicating a reduction in sampling uniformity. The sequential algorithm enhances uniformity by solving lots of maximum entropy problems. In contrast, our algorithm avoids computationally expensive optimization, resulting in significantly lower computational cost. The non-uniformity in our algorithm primarily stems from the approximation of the weight vector $\hat{\mathbf{w}}$. Therefore, developing a better estimator for $\hat{\mathbf{w}}$ could further enhance sampling uniformity.

In this paper, we do not focus primarily on the frequencies of specific graphs, which may be of greater interest in other contexts. Instead, we treat all graphs satisfying a given degree sequence as effectively equivalent. In practice, the number of such graphs may be extremely large, and we believe that generating a large number of distinct graphs may be more crucial than the uniformity of the sampling distribution in such cases. Our algorithm demonstrates satisfactory performance with respect to this consideration. In addition, we have evaluated algorithmic performance in terms of graph sampling speed, the number of unique graphs generated, and the uniformity of the sampling distribution in this paper. In future research, we plan to further explore additional evaluation metrics that more accurately capture algorithm performance from a practical perspective.

5 CONCLUSION

In this paper, we propose an efficient algorithm for reconstructing bipartite graphs with prescribed degree sequences. The algorithm starts by sorting the two degree sequences based on a specific rule. One sequence is grouped by degree values, and nodes in the other sequence are connected to the nodes in these groups.

Table 2: Comparison of the sampling uniformity between the EBGS algorithm and the sequential algorithm.

Degree Sequence		EBGS Algorithm			Sequential Algorithm		
a	**b**	N	CV	KL	N	CV	KL
$(3,3,3,2,1)$	$(5,3,2,1,1)$	27	2.95E-01	4.32E-02	27	7.40E-02	2.77E-03
$(3,2,2,2,2)$	$(4,3,1,1,1,1)$	685	4.97E-01	1.28E-01	685	3.68E-01	6.96E-02
$(5,5,4,4,2,2)$	$(5,5,5,3,2,1,1)$	873	5.60E-01	1.51E-01	886	4.26E-01	9.13E-02
$(5,5,4,4,3,2)$	$(5,5,5,3,3,2)$	1304	6.26E-01	1.83E-01	1318	4.88E-01	1.19E-01

The number of connections is determined by the theorem we present. These grouping and connection steps are iteratively applied until all nodes are processed. We evaluate the performance of the algorithm through numerical experiments, comparing its average runtime and sampling uniformity with those of the sequential algorithm.

Our current work focuses on improving and extending the proposed algorithm. One direction is to develop a better estimator for the weight vector, which directly impacts the uniformity of the sampling distribution. Another direction is to expand the algorithm's applicability beyond bipartite graphs. Since many real-world networks are not bipartite, we are exploring how to adapt the algorithm to support directed, undirected, and weighted graphs.

ACKNOWLEDGMENTS

This work was supported by the National Natural Science Foundation of China [Grants 72293562 and 72171060].

REFERENCES

Acharya, V. V., L. H. Pedersen, T. Philippon, and M. Richardson. 2016. "Measuring Systemic Risk". *The Review of Financial Studies* 30(1):2–47.

Ando, T., M. Greenwood-Nimmo, and Y. Shin. 2022. "Quantile Connectedness: Modeling Tail Behavior in the Topology of Financial Networks". *Management Science* 68(4):2401–2431.

Barvinok, A. 2010. "On the Number of Matrices and a Random Matrix with Prescribed Row and Column Sums and 0–1 Entries". *Advances in Mathematics* 224(1):316–339.

Caccioli, F., P. Barucca, and T. Kobayashi. 2018. "Network Models of Financial Systemic Risk: A Review". *Journal of Computational Social Science* 1(1):81–114.

Carstens, C. J., and K. J. Horadam. 2016, 10. "Switching edges to randomize networks: what goes wrong and how to fix it". *Journal of Complex Networks* 5(3):337–351.

Diaconis, P., and B. Sturmfels. 1998. "Algebraic Algorithms for Sampling from Conditional Distributions". *The Annals of Statistics* 26(1):363–397.

Fayle, T. M., and A. Manica. 2010. "Reducing Over-Reporting of Deterministic Co-Occurrence Patterns in Biotic Communities". *Ecological Modelling* 221(19):2237–2242.

Fosdick, B. K., D. B. Larremore, J. Nishimura, and J. Ugander. 2018. "Configuring Random Graph Models with Fixed Degree Sequences". *SIAM Review* 60(2):315–355.

Gale, D. 1957. "A Theorem on Flows in Networks". *Pacific Journal of Mathematics* 7(2):1073–1082.

Gandy, A., and L. A. M. Veraart. 2017. "A Bayesian Methodology for Systemic Risk Assessment in Financial Networks". *Management Science* 63(12):4428–4446.

Glasserman, P., and E. L. de Larrea. 2018. "Simulation of Bipartite or Directed Graphs with Prescribed Degree Sequences Using Maximum Entory Probabilities". In *2018 Winter Simulation Conference (WSC)*, 1658–1669 https://doi.org/10.1109/WSC.2018.8632479.

Glasserman, P., and E. L. de Larrea. 2023. "Maximum Entropy Distributions with Applications to Graph Simulation". *Operations Research* 71(5):1908–1924.

Huang, X., I. Vodenska, S. Havlin, and H. E. Stanley. 2013. "Cascading Failures in Bi-Partite Graphs: Model for Systemic Risk Propagation". *Scientific Reports* 3(1):1219.

Lux, T. 2016. "A Model of the Topology of the Bank – Firm Credit Network and Its Role as Channel of Contagion". *Journal of Economic Dynamics and Control* 66:36–53.

Verhelst, N. D. 2008. "An Efficient MCMC Algorithm to Sample Binary Matrices with Fixed Marginals". *Psychometrika* 73(4):705–728.

Wormald, N. C. 1999. "Models of Random Regular Graphs". In *Surveys in Combinatorics, 1999*, edited by J. D. Lamb and D. A. Preece, London Mathematical Society Lecture Note Series, 239–298. Cambridge University Press.

AUTHOR BIOGRAPHIES

TONG SUN is a Ph.D. student in the School of Management at Harbin Institute of Technology. He received his B.S. degree in Computational Finance from Harbin Institute of Technology in 2023. His research interests include rare-event simulation and graph simulation. His email address is 24B910002@stu.hit.edu.cn.

JIANSHU HAO is a Ph.D. candidate in the School of Management at Harbin Institute of Technology. His research interests include risk management, machine learning, and graph simulation. His email address is 20B910016@stu.hit.edu.cn.

ZHIYANG ZHANG is a lecturer in the School of Science at Harbin University of Science and Technology. He received his Ph.D. degree in Probability and Mathematical Statistics from the University of Chinese Academy of Sciences. His research interests include stochastic processes, stochastic analysis and SDE, etc. His email address is zzy198735@126.com.

GUANGXIN JIANG is a professor in the School of Management at Harbin Institute of Technology. He earned his Ph.D. degree in Applied Mathematics from Tongji University. His research interests include stochastic models and simulation, machine learning, financial engineering and risk management, FinTech, etc. His email address is gxjiang@hit.edu.cn and his website is https://homepage.hit.edu.cn/jiangguangxin (Corresponding author).