BalanSiNG: Fast and Scalable Generation of Realistic Signed Networks

Jinhong Jung
Seoul National University
jinhongjung@snu.ac.kr

Ha-Myung Park
Kookmin University
hmpark@kookmin.ac.kr

U Kang
Seoul National University
ukang@snu.ac.kr

ABSTRACT

How can we efficiently generate large-scale signed networks following real-world properties? Due to its rich modeling capability of representing trust relations as positive and negative edges, signed networks have spurred much interests with various applications. Despite its importance, however, existing models for generating signed networks do not correctly reflect properties of real-world signed networks.

In this paper, we propose BalanSiNG, a novel, scalable, and fully parallelizable method for generating large-scale signed networks following realistic properties. We identify a self-similar balanced structure observed from a real-world signed network, and simulate the self-similarity via Kronecker product. Then, we exploit noise and careful weighting of signs such that our resulting network obeys various properties of real-world signed networks. BalanSiNG is easily parallelizable, and we implement it using Spark. Extensive experiments show that BalanSiNG efficiently generates the most realistic signed networks satisfying various desired properties.

KEYWORDS

Signed Network Modeling; Balance Theory; Stochastic Kronecker Signed Graph; Balanced Signed Network Generator

1 INTRODUCTION

Signed networks [26] exhibit relationships between nodes as positive (trust) and negative (distrust) edges, and various online social services such as Epinions [10] have naturally formed signed networks by allowing users to express their trust. Inspired by these interesting trust relationships, many researchers have been recently attracted to mining useful information from signed networks, inducing advanced techniques for diverse applications such as sign prediction [22, 25], link prediction [40, 47], node ranking [15, 16, 28], node embedding [20, 46], node classification [42], anomaly detection [21], and community detection [5, 48].

Even though signed networks are important resources in social network analysis, the understanding of synthetically generating realistic signed networks from scratch was nascent. In unsigned networks, many sophisticated generation models have been proposed, including Barabási-Albert (BA) [1], Forest Fire (FF) [27], Stochastic Kronecker Graph (SKG) [23], and Recursive Matrix (R-MAT) [4]. Among those models, SKG and R-MAT have received significant interest from data mining communities [12, 13, 31, 34, 37] since they well capture various properties of real-world graphs such as power-law degree distributions [1, 8, 9, 23, 31], shrinking effective diameters [3, 9, 27], power-law singular value distribution [4, 9, 23], etc.

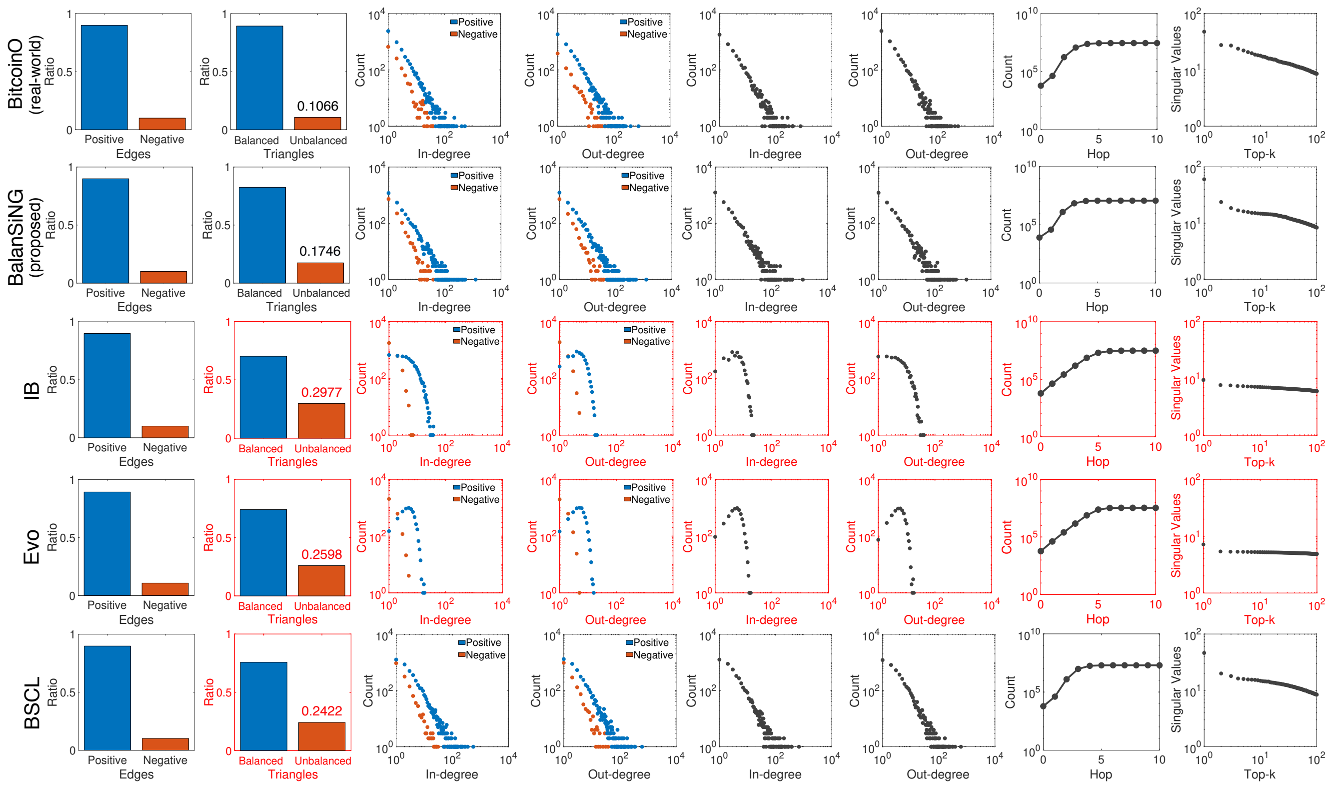
However, the existing models cannot generate realistic signed networks because they do not provide a mechanism for determining signs of edges. Real-world signed networks exhibit not only the traditional properties in unsigned networks, but also distinct characteristics derived from signs (Figure 11). Especially, real-world signed networks are dominated under balance theory [2, 11] that plays a crucial role in the construction of signed networks [6, 26]. According to the balance theory, balanced triangles are more likely to be created than unbalanced ones in real signed networks (details in Section 2). Thus, modeling signed networks demands careful considerations on how to positively or negatively associate three nodes on each triangle.

Motivated by this, several methods have been proposed for signed network generation considering the balance theory. Vukašinović et al. [45] proposed an interaction based model (IB) simulating the generation of signed edges using ant pheromone mechanism and the balance theory. Ludwig et al. [29] suggested an evolutionary model (Evo) that randomly inserts or removes signed edges over time so that the evolving network follows the balance theory. Derr et al. [6] have recently proposed Balanced Signed Chung-Lu (BSCL), the state-of-the-art model imitating an input network based on Transitive Chung-Lu [35] and the balance theory. However, they are limited in generating realistic signed networks (see Figure 1), and computationally inefficient. Furthermore, the scale of existing signed networks remains small; consequently, researchers have suffered from the lack of large-scale signed networks when testing the scalability of their methods. Thus, generating realistic large-scale networks is extremely useful to evaluate the scalability [14, 19, 30–32], simulate their performance depending on various properties of networks [17, 18, 23, 39], and anonymize their data [6, 24].

In this paper, we propose BalanSiNG (Balanced Signed Network Generator), a novel and scalable method for generating synthetic but realistic signed networks. We first identify a self-similar pattern observed from a real signed network. Then, we design Basic Stochastic Kronecker Signed Graph (SKSG-B), a basic model that simulates the self-similarity using Kronecker product and generates fully balanced signed networks. On top of SKSG-B, we propose Stochastic Kronecker Signed Graph (SKSG) by adding random noises to the self-similar pattern and introducing careful weighting to increase the probability of forming positive edges to generate signed networks following real-world properties. From SKSG, we derive BalanSiNG that efficiently creates signed edges fully in parallel. Through extensive experiments, we show that BalanSiNG efficiently generates the most realistic signed networks capturing various properties of real-world signed networks.

Our main contributions are summarized as follows:

- **Novel self-similarity.** We suggest a novel self-similar pattern called *self-similar balanced structure* to be satisfied for generating signed networks (Figure 4).
- **Method.** We propose BalanSiNG, an efficient and parallel method that simulates the suggested self-similarity



(a) Global sign distribution (b) Balanced triangle distribution (c) Signed in-degree distribution (d) Signed out-degree distribution (e) In-degree distribution (f) Out-degree distribution (g) Hop plot (h) Singular value distribution

Figure 1: BalanSiNG generates the most similar network to the real-world network, compared to other methods. The plots show the comparison of properties from real-world signed networks and those from BalanSiNG and competitors. We use BitcoinO dataset [22] for representing the properties of real-world signed networks; other real-world networks give similar results. (a)-(d) illustrate properties derived from edge signs, and (e)-(h) depict traditional properties of real-world networks regardless of edge signs (see Section 2.1). Red colored boxes denote that the corresponding graph does not match the corresponding property.

by using Kronecker product, exploiting noise, and careful weighting (Algorithm 3). BalanSiNG generates signed networks satisfying various desired properties of real-world signed networks listed in Section 2.1.

- **Experiments.** We demonstrate that BalanSiNG generates the most realistic signed networks following real-world properties compared to competitors as shown in Figure 1. We also show that BalanSiNG generates signed networks up to 265× faster than the state-of-the-art method, and near linearly scales up w.r.t. the number of edges on both single and distributed machines (Figure 9).

The source code of BalanSiNG and datasets are available at **https://datalab.snu.ac.kr/balansing**.

2 PRELIMINARIES

We describe the desired properties of real-world signed networks to consider when generating a synthetic signed network in Section 2.1. We formally define the problem addressed in this paper in Section 2.2. We then review Stochastic Kronecker Graph (SKG), a representative generation model for unsigned networks to capture the concept of self-similarity simulation in Section 2.3.

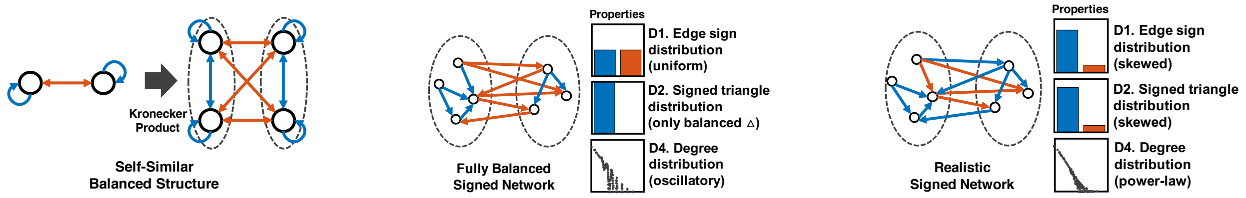
Symbols used in this paper are summarized in Table 1. Throughout the paper, we use a blue arrow and a red arrow to indicate a positive edge and a negative edge, respectively.

2.1 Desired Properties of Signed Networks

We investigate real-world signed networks to grasp their unique properties to be satisfied when generating signed networks. As shown in the first row of Figure 1, there are not only unique properties derived from signs on edges but also traditional ones studied in unsigned networks. The properties of other real-world networks are in Figure 11. We examine properties induced by signs in Section 2.1.1, and then review the typical ones regardless of signs in Section 2.1.2.

2.1.1 *Properties with respect to signs on edges.*

- **D1) Positively skewed sign proportion [25, 26, 43].** Real-world signed networks contain much more positive edges than negative ones, as demonstrated in the first row of Figure 1(a).
- **D2) Highly balanced triangle proportion [6, 11, 26, 41, 43, 48].** Signed triangles have been extensively studied in signed networks based on balance theory [2, 11] stating that triangles $\triangle_{+++}$ with three positive signs and those $\triangle_{+--}$ with one positive sign are much more plausible than other types of triangles $\triangle_{++-}$ and $\triangle_{---}$. The former are called *balanced triangles*, and the latter are *unbalanced triangles*. Thus, the ratio of balanced triangles is much larger than that of unbalanced triangles as shown in Figure 1(b).
- **D3) Power-law degree distribution for only positive or negative edges [43].** In scale-free networks, in- and out-degree distributions follow a power-law [1]. In real-world signed networks, when we consider only positive (or negative) edges, corresponding degree distributions also follow power-laws as shown in Figures 1(c) and 1(d).



(a) Self-similar balanced structure to be simulated by Kronecker product

(b) Signed network following Basic Stochastic Kronecker Signed Graph (SKSG-B)

(c) Signed network following Stochastic Kronecker Signed Graph (SKSG)

Figure 2: Overview of our approach. (a) We suggest *self-similar balanced structure* observed from a real-world signed network (Section 3.1). (b) We design SKSG-B, a basic version of our model, that simulates the self-similarity using Kronecker product, and generates a fully balanced signed network (Section 3.2). (c) We then propose SKSG, an advanced version that produces realistic signed networks by introducing noise and weight splitting (Section 3.3). From SKSG model, we derive BalanSiNG which quickly generates realistic signed networks satisfying the desired properties in parallel (Section 3.4).

2.1.2 Properties without respect to signs on edges.

- **D4) Power-law degree distribution [1, 8, 9, 23, 31].** Real-world networks without signs also show power-law degree distributions as shown in Figures 1(e) and 1(f).
- **D5) Small effective diameter (hop plot) [3, 9, 27].** The hop plot shows the ratio of node pairs reachable from each other within k-hop for each integer k. It is closely related to the effective diameter, the 90 percentile distance in the hop plot. As seen in Figure 1(g), the effective diameters of real-world graphs are small (typically between 4 and 5).
- **D6) Power-law singular value distribution [4, 9, 23].** The singular values in the adjacency matrix of a real graph follow a power-law distribution as shown in Figure 1(h).

2.2 Problem Definition

Problem 1 (Signed Network Generation). *Given the target numbers* $|\mathrm{V}|$ *and* $|\mathrm{E}|$ *of nodes and edges, respectively, we aim to synthetically generate a directed signed network from scratch having* $|\mathrm{V}|$ *nodes and* $|\mathrm{E}|$ *signed edges where the output network should follow the desired properties of real-world signed networks listed in Section 2.1.* ■

2.3 SKG: Stochastic Kronecker Graph Model

SKG [23] is an unsigned network generation model based on Kronecker product. Its motivation is that power-law phenomena in nature occur due to self-similarity, i.e., a self-similar object is approximately similar to a part of itself [36]. SKG stochastically simulates a self-similarity with a tiny seed graph using Kronecker product denoted by $\otimes$ (Definition D.1 in Appendix D). Specifically, SKG creates a self-similar graph by recursively computing $\mathcal{A}^{(k)} = \mathcal{A}_{\text{seed}} \otimes \mathcal{A}^{(k-1)}$ where $\mathcal{A}^{(k)}$ is k-th Kronecker product result over the adjacency matrix $\mathcal{A}_{\text{seed}}$ of the seed graph. In SKG, (u, v)-th entry of $\mathcal{A}^{(k)}$ is the probability $P(u, v)$ that edge $u \to v$ exists in the graph. When a randomly generated value for each entry is within the probability, the corresponding edge is created. Several methods such as FastKronecker [24] and R-MAT [4] were proposed to reduce the generation time of SKG.

Although many research works [23, 37] have shown that SKG well captures various real-world properties (e.g., D4-6) in unsigned networks, the model is not proper for modeling signed networks since it does not consider how to form signs on edges. More essentially, it has not been revealed which self-similarity should be simulated when we generate signed networks through Kronecker products. Hence, our main challenge is to identify a desirable self-similarity for generating signed networks based on Kronecker product so that a resulting network establishes a solid foundation for the aforementioned properties.

Table 1: Table of symbols.

Symbol	Definition
V	set of nodes
E	set of singed edges
$\otimes$	Kronecker product
L	target recursion depth
$\mathbf{T}$	stochastic signed tensor $\mathbf{T} \in \mathbb{R}^{\lvert\mathrm{V}\rvert \times \lvert\mathrm{V}\rvert \times 2} = \{+\mathcal{P}, -\mathcal{M}\}$
$\mathbf{T}_{\text{seed}}$	$2 \times 2 \times 2$ seed stochastic signed adjacency tensor, i.e., $\mathbf{T}_{\text{seed}} = \{+\mathcal{P}_{\text{seed}}, -\mathcal{M}_{\text{seed}}\}$
$\mathbf{N}_{\text{seed}}^{(l)}$	$2 \times 2 \times 2$ seed tensor with noise at level l
$f_b(\cdot)$	balanced sign aggregator in Definition 3.2
$f_\alpha(\cdot)$	weight splitter with α in Definition 3.4
$\tilde{\mathbf{T}}^{(l)}$	l-th Kronecker product result with $f_b(\cdot)$ and $f_\alpha(\cdot)$
γ	parameter for noise
α	parameter for weight splitting
$\rho(\cdot)$	ratio of a given input
$\triangle_b$ and $\triangle_u$	balanced and unbalanced triangles, respectively
$\triangle_{+++}$	balanced triangles with three $+$ signs
$\triangle_{++-}$	unbalanced triangles with two $+$ signs and one $-$ sign
$\triangle_{+--}$	balanced triangles with one $+$ sign and two $-$ signs
$\triangle_{---}$	unbalanced triangles with three $-$ signs

3 PROPOSED METHOD

We propose BalanSiNG, a novel method for generating realistic signed networks following the desired properties in Section 2.1. The technical challenges and our approaches are as follows:

- Which self-similarity should be satisfied for generating signed networks (Section 3.1)? We suggest a novel self-similarity called *self-similar balanced structure* to be satisfied for generating balanced signed networks by investigating a real-world signed network.
- How can we generate signed networks following the self-similarity (Section 3.2)? We design Basic Stochastic Kronecker Signed Graph (SKSG-B), a basic model that produces a fully balanced signed network by simulating the self-similarity via Kronecker product.
- How can we generate realistic signed networks (Section 3.3)? We propose Stochastic Kronecker Signed Graph (SKSG), an advanced model introducing noise and weight splitting to SKSG-B so that the resulting network exhibits the aforementioned characteristics in Section 2.1.
- How can we efficiently generate large-scale signed networks (Section 3.4)? We derive Balanced Signed Network Generator (BalanSiNG) from SKSG, a fully parallelizable method that quickly generates signed edges.

We illustrate the overview of our approaches in Figure 2. Our main goal is to design a generation method for signed networks showing the distinct properties of real world signed networks. Among the various properties, we mainly focus on the balanced

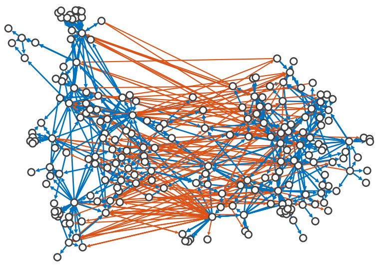


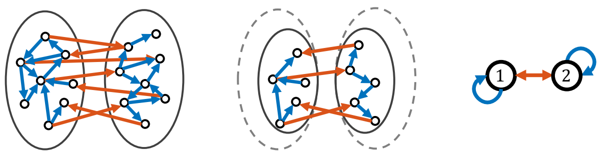
Figure 3: Balanced Structure in Congress dataset [44] where two large clusters are observed. Most nodes in each cluster are positively connected, and nodes between the clusters are negatively connected.

triangle distribution indicating *balanced signed networks* since it is one of the most distinct properties derived from signs [6, 26]. For that purpose, we first design a self-similarity to be satisfied for balanced signed networks, inspired from balanced structure in signed networks as shown in Figure 2(a).

We then propose a basic model SKSG-B and an advanced model SKSG. SKSG-B simulates the self-similarity using Kronecker product so that it produces a fully balanced signed network (i.e., there are no unbalanced triangles) as depicted in Figure 2(b). However, we observe that the fully balanced network of SKSG-B has different properties than those from real-world signed networks in terms of edge sign and balanced triangle proportions as shown in Figure 2(b). Thus, we suggest SKSG by introducing noise and weight splitting to SKSG-B so that SKSG produces realistic signed networks following the desired properties as seen in Figure 2(c). Furthermore, we develop BalanSiNG that generates balanced signed networks fully in parallel while supporting SKSG.

3.1 Self-Similarity for Signed Networks

We investigate a real-world signed network to understand its structure with signs, and then model a self-similarity behind the structure. We analyze the Congress dataset [44], a real-world signed network where nodes represent politicians, and signed edges indicate supports (i.e., positive) or oppositions (i.e., negative) between nodes. The detailed statistics of the dataset are summarized in Table 4. We visualize the signed network of the Congress dataset in Figure 3. Note that two distinct clusters appear where most nodes are mutually friends in each cluster while nodes between the clusters exhibit mutual antagonism. This structure is called *balanced signed network*. If a signed network is *fully balanced* [7], there are *two groups*; nodes in each group create only positive edges while nodes between the groups form only negative edges as in Figure 4(a). This structure is directly related to balance theory [11] since there are only balanced triangles in a fully balanced network, i.e., there are only triangles $\triangle_{+++}$ in each group and triangles $\triangle_{+--}$ between the groups.



(a) Global balanced structure (b) Zoomed-in balanced structure (c) Self-similar balanced structure

Figure 4: Self-similar pattern in balanced signed networks.

From this structure, we observe a self-similar pattern, called *self-similar balanced structure*, as illustrated in Figure 4. Figure 4(a) represents a fully balanced signed network. Then, if we zoom in the network as in Figure 4(b), a smaller but similar structure to that of Figure 4(a) appears. Note that the structure in Figure 4(b) is also balanced; hence, the balanced structure is self-similar according to the definition of self-similarity [36]. We abstract the self-similar balanced structure as shown in Figure 4(c) where each node indicates a group, and blue edges represent that positive edges are created within each group while red edges indicate that negative edges are formed between the groups.

Algorithm 1: SKSG-B

Input: seed tensor $\mathbf{T}_{\text{seed}} \in \mathbb{R}^{2\times2\times2}$, target recursion level L, and target number $|\mathrm{E}|$ of edges
Output: set E of signed edges
1: set $\tilde{\mathbf{T}}^{(1)} \leftarrow \mathbf{T}_{\text{seed}}$ and $\mathrm{E} \leftarrow \emptyset$
2: **for** $l = 2$ to L **do**
3: compute $\tilde{\mathbf{T}}^{(l)} \leftarrow f_b(\tilde{\mathbf{T}}^{(1)} \otimes \tilde{\mathbf{T}}^{(l-1)})$ in Equation (4)
4: **for each** (u, v) such that $u, v \in \mathrm{V}$ **do**
5: set $P(u, v, s) \leftarrow \tilde{\mathbf{T}}^{(L)}_{uvs}$ where $s \in \{+, -\}$
6: compute $P(u, v)$ and $P(s|u, v)$ using Equation (2)
7: toss a biased coin with $P(u, v)$
8: **if** head appears, i.e., $u \rightarrow v$ is formed **then**
9: $\hat{s} \leftarrow \mathrm{argmax}_{s\in\{+,-\}} P(s|u, v)$
10: insert a signed edge $(u \rightarrow v, \hat{s})$ into E if $|\mathrm{E}| < m$
11: **return** set E of signed edges

3.2 SKSG-B: Basic Stochastic Kronecker Signed Graph Model

We describe our basic model SKSG-B for modeling signed networks. The main intuition of SKSG-B is to simulate the self-similarity explained in Section 3.1 using Kronecker product.

3.2.1 Formulation of SKSG-B. First of all, we define *stochastic signed tensor* used for constructing a signed network G as follows:

Definition 3.1 (Stochastic Signed Tensor). Let $|\mathrm{V}|$ be the number of nodes. A stochastic signed tensor $\mathbf{T} \in \mathbb{R}^{|\mathrm{V}|\times|\mathrm{V}|\times2}$ consists of two stochastic adjacency matrices $\mathcal{P} \in \mathbb{R}^{|\mathrm{V}|\times|\mathrm{V}|}$ and $\mathcal{M} \in \mathbb{R}^{|\mathrm{V}|\times|\mathrm{V}|}$ with signs, i.e., $\mathbf{T} = \{+\mathcal{P}, -\mathcal{M}\}$ where $\mathcal{P}$ and $\mathcal{M}$ represent probabilities for positive and negative edges, respectively. ■

Then, the self-similar balanced structure in Figure 4(c) is represented as follows:

$$\mathbf{T}_{\text{seed}} = \{+\mathcal{P}_{\text{seed}}, -\mathcal{M}_{\text{seed}}\} = \left\{+\begin{bmatrix} p_{11} & 0 \\ 0 & p_{22} \end{bmatrix}, -\begin{bmatrix} 0 & m_{12} \\ m_{21} & 0 \end{bmatrix}\right\} \quad (1)$$

where + and − indicate positive and negative signs, respectively. Each entry $\mathbf{T}_{uvs}$ is a joint probability $P(u, v, s)$ where u and v are nodes, and $s \in \{+, -\}$ is a sign, e.g., $\mathbf{T}_{12-} = m_{12} = P(1, 2, -)$. The sum of all $P(u, v, s)$ is 1, i.e., $\sum_{(u,v,s)} P(u, v, s) = 1$. If we know $P(u, v, s)$, we are able to determine the creation process of edge $u \rightarrow v$ and its sign. First, we compute $P(u, v) = P(u, v, +) + P(u, v, -)$, toss a biased coin with $P(u, v)$, and determine to create the edge if the coin's head appears (line 7 in Algorithm 1). If $u \rightarrow v$ is formed, we decide its sign based on $P(s|u, v)$ as follows:

$$P(s|u, v) = \frac{P(u, v, s)}{\sum_{t\in\{+,-\}} P(u, v, t)} = \frac{P(u, v, s)}{P(u, v)} \quad (2)$$

If $P(+|u, v) > P(-|u, v)$, then its sign is determined to be positive, otherwise, it is negative (line 9 in Algorithm 1). Note that we call this approach *deterministic sign decision*.

Given a small seed signed tensor $\mathbf{T}_{\text{seed}}$, SKSG-B repeats Kronecker product multiple times over $\mathbf{T}_{\text{seed}}$. Kronecker product between two signed tensors is defined as follows:

$$\begin{aligned} \mathbf{T}^{(2)} = \mathbf{T}^{(1)} \otimes \mathbf{T}^{(1)} &= \{+\mathcal{P}, -\mathcal{M}\} \otimes \{+\mathcal{P}, -\mathcal{M}\} \\ &= \{+\mathcal{P} \otimes \mathcal{P}, -\mathcal{P} \otimes \mathcal{M}, -\mathcal{M} \otimes \mathcal{P}, +\mathcal{M} \otimes \mathcal{M}\} \end{aligned} \quad (3)$$

where $\mathbf{T}^{(k)}$ is k-th Kronecker product result on $\mathbf{T}_{\text{seed}} = \mathbf{T}^{(1)} \in \mathbb{R}^{n\times n\times2}$. Note that the dimension of $\mathbf{T}^{(2)}$ is $n^2 \times n^2 \times 2^2$ where the last dimension indicates $\{++, +-, -+, --\}$. Each entry of $\mathbf{T}^{(2)}$ indicates a joint probability $P(u, v, \{s, s\})$. However, this is not the probability that we want since we need $P(u, v, s)$ to determine the edge's sign. Hence, we aggregate the terms according to their sign using *balanced sign aggregator* $f_b(\cdot)$ defined as follows:

Definition 3.2 (Balanced Sign Aggregator). Balanced sign aggregator $f_b : \mathbb{R}^{N\times N\times 4} \to \mathbb{R}^{N\times N\times 2}$ aggregates the terms in Equation (3) according to their signs as follows:

$$\begin{aligned}\tilde{\mathbf{T}} = f_b(\mathbf{T}\otimes\mathbf{T}) &= \{+(\mathcal{P}\otimes\mathcal{P} + \mathcal{M}\otimes\mathcal{M}), -(\mathcal{P}\otimes\mathcal{M} + \mathcal{M}\otimes\mathcal{P})\}\\ &= \{+\tilde{\mathcal{P}}, -\tilde{\mathcal{M}}\}\end{aligned}$$

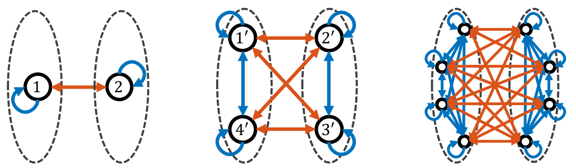
where $\tilde{\mathbf{T}} \in \mathbb{R}^{N\times N\times 2}$ is a signed tensor aggregated by $f_b(\cdot)$, $\tilde{\mathcal{P}} = \mathcal{P}\otimes\mathcal{P} + \mathcal{M}\otimes\mathcal{M}$, and $\tilde{\mathcal{M}} = \mathcal{P}\otimes\mathcal{M} + \mathcal{M}\otimes\mathcal{P}$. ■

The Kronecker product result with $f_b(\cdot)$ is guaranteed to form a fully balanced signed network (see Section 3.2.2 and Lemma 3.3). Let $\tilde{\mathbf{T}}^{(l)}$ denote l-th Kronecker product result with $f_b(\cdot)$, and $\tilde{\mathbf{T}}^{(1)}$ is initially set to $\mathbf{T}_{\text{seed}}$ in Equation (1). Then, we generalize the Equation (3) as follows:

$$\tilde{\mathbf{T}}^{(l)} = f_b(\tilde{\mathbf{T}}^{(1)} \otimes \tilde{\mathbf{T}}^{(l-1)}) \tag{4}$$

where $\tilde{\mathbf{T}}^{(l)} \in \mathbb{R}^{n^l\times n^l\times 2}$ is used for building a signed network G given the recursion level l. Algorithm 1 summarizes SKSG-B based on Equation (4). Given $\mathbf{T}_{\text{seed}}$ in Equation (1), a target recursion level L, and a number $|\mathrm{E}|$ of edges, SKSG-B generates a signed network having 2^L nodes and $|\mathrm{E}|$ edges (line 3). For each pair of nodes, it decides the creation of the edge (line 7) and its sign (line 9) based on $\tilde{\mathbf{T}}^{(L)}$.

3.2.2 Self-similar Balanced Network Simulated by SKSG-B. We illustrate how SKSG-B simulates the self-similarity for balanced signed networks. Given $\tilde{\mathbf{T}}^{(1)} = \mathbf{T}_{\text{seed}}$ in Equation (1), we compute $\tilde{\mathbf{T}}^{(2)}, \tilde{\mathbf{T}}^{(3)}, \cdots$ based on Equation (4). Figure 5 depicts the results of $\tilde{\mathbf{T}}^{(1)}$, $\tilde{\mathbf{T}}^{(2)}$ and $\tilde{\mathbf{T}}^{(3)}$. Note that the balanced structure is kept as level l increases, i.e., only positive edges are formed within each group (dotted ellipses), and only negative edges are allowed between the groups when we start from $\mathbf{T}_{\text{seed}}$ in Equation (1).



(a) $\tilde{\mathbf{T}}^{(1)}$ (b) $\tilde{\mathbf{T}}^{(2)}$ (c) $\tilde{\mathbf{T}}^{(3)}$

Figure 5: Illustrations on how SKSG-B simulates the self-similarity for balanced signed networks given $\mathbf{T}_{\text{seed}} = \tilde{\mathbf{T}}^{(1)}$.

We formalize this property of the balanced structure generated by SKSG-B in the following lemma:

LEMMA 3.3. *Given* $\mathbf{T}_{\text{seed}}$ *in Equation* (1), $\tilde{\mathbf{T}}^{(l)}$ *of Equation* (4) *produces a fully balanced signed network.*

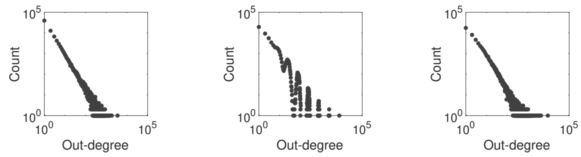
PROOF. See the detailed proof in Appendix B. □

3.3 SKSG: Exploiting Noise and Weight for Realistic Signed Networks

We propose STOCHASTIC KRONECKER SIGNED GRAPH (SKSG), an advanced model from SKSG-B for generating signed networks following the desired properties in Section 2.1. Although SKSG-B simulates a fully balanced signed network, we will observe that the network's properties deviate from those of real signed networks. We explain the issues of SKSG-B step by step, and suggest how to resolve each issue in the following subsections. The approaches of SKSG are summarized in Algorithm 2.

Algorithm 2: SKSG

Input: seed tensor $\mathbf{T}_{\text{seed}} \in \mathbb{R}^{2\times 2\times 2}$, target recursion level L, target number $|\mathrm{E}|$ of edges, noise parameter γ, and weight parameter α
Output: set E of signed edges
1: generate random noises $\mu^{(l)} \in [-\gamma, \gamma]$ [37], and obtain noisy seed tensors $\mathbf{N}^{(l)}_{\text{seed}}$ using Equation (6) with $\mathbf{T}_{\text{seed}}$ and $\mu^{(l)}$ for $1 \le l \le L$
2: set $\tilde{\mathbf{T}}^{(1)} \leftarrow \mathbf{N}^{(1)}_{\text{seed}}$ and $\mathrm{E} \leftarrow \emptyset$
3: **for** $l = 2$ to L **do**
4: compute $\tilde{\mathbf{T}}^{(l)} \leftarrow f_\alpha(f_b(\mathbf{N}^{(l)}_{\text{seed}} \otimes \tilde{\mathbf{T}}^{(l-1)}))$ in Equation (7)
5: **for each** (u, v) such that $u, v \in \mathrm{V}$ **do**
6: set $P(u, v, s) \leftarrow \tilde{\mathbf{T}}^{(L)}_{uvs}$ for $s \in \{+, -\}$
7: compute $P(u, v)$ and $P(s|u, v)$ using Equation (2)
8: toss a biased coin with $P(u, v)$
9: **if** head appears, i.e., $u \to v$ is formed **then**
10: toss a biased coin with $P(+|u, v)$
if head appears, **then** $\hat{s} \leftarrow +$, **otherwise**, $\hat{s} \leftarrow -$
11: insert a signed edge $(u \to v, \hat{s})$ into E if $|\mathrm{E}| < m$
12: **return** set E of signed edges



(a) Real network (b) SKSG-B without noises (c) SKSG with noises

Figure 6: Out-degree distributions of (a) the Epinions dataset, (b) a network from SKSG-B, and (c) a network from SKSG.

3.3.1 Introducing Noise (line 1 in Algorithm 2). We investigate whether a degree distribution of a graph from SKSG-B follows a power-law. We focus on degree distributions regardless of edge signs (i.e., D4). For $\mathbf{T}_{\text{seed}}$, we use the values of Equation (9) in Section 4.1.3. Figure 6(b) shows the out-degree distribution of SKSG-B. Note that the distribution exhibits oscillations; it is far from being monotonically decreasing unlike that of real networks as in Figure 6(a). In fact, the oscillatory behavior is a well-known issue of the standard SKG [37]. Since the edge formation of SKSG-B is equivalent to that of SKG (see the details in Appendix E), SKSG-B naturally inherits the oscillatory behavior from SKG.

Seshadhri et al. [37] analyzed the oscillatory issue of SKG, and provided a technique called Noisy SKG. For each level l, Noisy SKG defines a noise seed matrix $\mathcal{A}^{(l)}_{\text{seed}} \in \mathbb{R}^{2\times 2}$ by introducing a random noise $\mu^{(l)}$ to the seed matrix $\mathcal{A}_{\text{seed}} \in \mathbb{R}^{2\times 2}$. More specifically, $\mu^{(l)}$ is chosen uniformly at random in $[-\gamma, \gamma]$ for $\gamma \le \min(\frac{a_{11}+a_{22}}{2}, a_{12})$ while a_{ij} denotes (i, j)-th entry of $\mathcal{A}_{\text{seed}}$ defined as follows:

$$\mathcal{A}^{(l)}_{\text{seed}} = \begin{bmatrix} a_{11} - \frac{2\mu^{(l)}a_{11}}{a_{11}+a_{22}} & a_{12} + \mu^{(l)} \\ a_{21} + \mu^{(l)} & a_{22} - \frac{2\mu^{(l)}a_{22}}{a_{11}+a_{22}} \end{bmatrix} \tag{5}$$

Note that its entries sum to 1, and the expectation of $\mathcal{A}^{(l)}_{\text{seed}}$ is $\mathcal{A}_{\text{seed}}$. This approach introduces randomness to the degree of each node so that the fluctuation in the degree distribution is removed, which is theoretically and empirically proved in [37, 38].

In this work, we adopt this technique to our advanced model SKSG for power-law degree distributions in its signed networks. We aim to obtain a noisy seed tensor $\mathbf{N}^{(l)}_{\text{seed}}$ by adding a noise $\mu^{(l)}$ to the seed tensor $\mathbf{T}_{\text{seed}} = \{+\mathcal{P}_{\text{seed}}, -\mathcal{M}_{\text{seed}}\}$ of Equation (1) for each level l as follows (line 1 in Algorithm 2):

$$\mathrm{N}_{\text{seed}}^{(l)} = \{+\mathcal{P}_{\text{seed}}^{(l)}, -\mathcal{M}_{\text{seed}}^{(l)}\} \tag{6}$$
$$= \left\{ + \begin{bmatrix} p_{11} - \frac{2\mu^{(l)} p_{11}}{p_{11}+p_{22}} & 0 \\ 0 & p_{22} - \frac{2\mu^{(l)} p_{22}}{p_{11}+p_{22}} \end{bmatrix}, - \begin{bmatrix} 0 & m_{12} + \mu^{(l)} \\ m_{21} + \mu^{(l)} & 0 \end{bmatrix} \right\}$$

where $\mu^{(l)}$ is a uniform random noise selected in $[-\gamma, \gamma]$ for $\gamma \leq \min(\frac{p_{11}+p_{22}}{2}, m_{12})$. Note that Equation (6) is derived from Equation (5) such that $\mathcal{A}_{\text{seed}}^{(l)} = \mathcal{P}_{\text{seed}}^{(l)} + \mathcal{M}_{\text{seed}}^{(l)}$, while preserving the self-similar balanced structure in Equation (1). Thus, our approach is able to model the probability of edge sign as well as the randomness of node degree while Noisy SKG with Equation (5) cannot model the probability for deciding the sign of an edge.

When generating a signed edge, we exploit $\mathrm{N}_{\text{seed}}^{(l)}$ according to level l instead of the original $\mathbf{T}_{\text{seed}}$ as in line 4 of Algorithm 2 (see Equation (7) in Section 3.3.2). Figure 6(c) depicts the out-degree distribution of SKSG using $\mathrm{N}_{\text{seed}}^{(l)}$ with $\gamma = 0.1$. The in-degree distribution of SKSG also shows the similar tendency.

3.3.2 Weight Splitting (line 4 in Algorithm 2). We analyze the properties about signs in a network of SKSG-B. As shown in Table 2, the ratio of positive edges in a network of SKSG-B is almost equal to that of negative ones, and there are only balanced triangles because SKSG-B generates fully balanced signed networks. However, real signed networks exhibit positively skewed sign and highly balanced triangle proportions (i.e., there are few unbalanced triangles) as seen in the 'BitcoinO' column of Table 2.

Table 2: Sign and balanced triangle ratios. $\rho(+)$ and $\rho(-)$ are the ratios of positive and negative edges, respectively. $\rho(\triangle_b)$ and $\rho(\triangle_u)$ are the ratios of balanced and unbalanced triangles, respectively.

	BitcoinO	SKSG-B	SKSG
$\rho(+)$	0.8999	0.5001	0.8993
$\rho(-)$	0.1001	0.4999	0.1007
$\rho(\triangle_b)$	0.8934	1.0000	0.8254
$\rho(\triangle_u)$	0.1066	0.0000	0.1746

To alleviate this issue, we suggest a weight splitting technique that increases the probabilities on generating positive signs. Note that SKSG-B produces a larger number of negative edges than expected; hence, we move a proportion of probabilities for negative signs into that for positive signs using the following function:

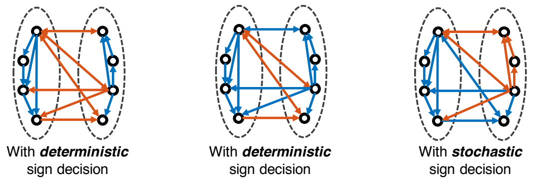
Definition 3.4 (Weight Splitter). For $0 < \alpha < 1$, weight splitter $f_\alpha : \mathbb{R}^{N \times N \times 2} \to \mathbb{R}^{N \times N \times 2}$ is defined as follows:

$$f_\alpha(\mathrm{T}) = f_\alpha(\{+\mathcal{P}, -\mathcal{M}\}) = \{+(\mathcal{P} + \alpha\mathcal{M}), -(1-\alpha)\mathcal{M}\} \quad \blacksquare$$

The function f_α increases the probabilities $\mathcal{P}$ of positive signs by $\alpha\mathcal{M}$. Equation (4) is extended with f_α and $\mathrm{N}_{\text{seed}}^{(l)}$ as follows:

$$\tilde{\mathbf{T}}^{(l)} = f_\alpha(f_b(\mathrm{N}_{\text{seed}}^{(l)} \otimes \tilde{\mathbf{T}}^{(l-1)})) \tag{7}$$

where $\tilde{\mathbf{T}}^{(l)}$ is the level-l result with f_α and f_b. The effects of f_α are that it 1) increases the proportion of positive edges, and 2) forms a few unbalanced triangles $\triangle_{++-}$ as in Figure 7(b). The reason for the latter is as follows. SKSG-B produces a fully balanced network; thus, there are two groups as in Figure 7(a). Since f_α decreases probabilities of negative sign by $\alpha\mathcal{M}$ at each level l in Equation (7), a negative edge between the groups in SKSG-B could become positive in SKSG, resulting in $\triangle_{++-}$ as in Figure 7(b). Note that after $f_\alpha(\cdot)$, probabilities for positive signs do not decrease (Definition 3.4); thus, SKSG with f_α still produces only positive edges inside a group if the edge sign is decided deterministically.



(a) SKSG-B without f_α (b) SKSG with f_α (c) SKSG with f_α

Figure 7: Effects of weight splitter f_α with (b) deterministic sign decision (Section 3.3.2) and (c) stochastic sign decision (Section 3.3.3).

3.3.3 Stochastic Sign Decision (line 10 in Algorithm 2). SKSG-B deterministically decides the sign of an edge based on $P(+|u, v) > P(-|u, v)$ (line 9 in Algorithm 1). However, this approach incurs a subtle issue: such decision does not produce $\triangle_{---}$ at all even though in real signed networks, there are a very few $\triangle_{---}$ as shown in Table 5. Although we introduce f_α in Section 3.3.2, SKSG with f_α does not form $\triangle_{---}$ since it generates only positive edges inside a group while for the case in Figure 7(b), the formation of $\triangle_{---}$ needs one negative edge inside a group with two negative ones between the groups. To resolve this issue, we suggest *stochastic sign decision* where SKSG stochastically decides the edge sign by tossing a biased coin with $P(+|u, v)$ (line 10 in Algorithm 2). This allows an edge to become negative with a low probability inside a group; thus, a few $\triangle_{---}$ are formed as in Figure 7(c). Based on $\tilde{\mathbf{T}}^{(L)}$ with f_α and stochastic sign decision, SKSG introduces the skewness of the sign and balanced triangle ratios similarly to those of the real network as shown in Table 2 where we use $\gamma = 0.1$, $\alpha = 0.75$, $L = 13$ and $\mathbf{T}_{\text{seed}}$ in Equation (9).

3.4 BalanSiNG: Fast and Scalable Balanced Signed Network Generator

We propose BalanSiNG, an efficient method for generating signed edges in parallel, while supporting SKSG. Algorithm 2 of SKSG is not scalable since its time and space complexities are $O(|V|^2)$, respectively, where $|V|$ is the number of nodes to be generated. The reason is that SKSG explicitly constructs signed tensor $\tilde{\mathbf{T}}^{(L)} \in \mathbb{R}^{2^L \times 2^L \times 2}$ through Kronecker product. Our main intuition to design a scalable method for the problem is to directly determine edge and track its sign probabilities without constructing $\tilde{\mathbf{T}}^{(L)}$ explicitly.

We summarize BalanSiNG in Algorithm 3. At each iteration, it exploits Generate-Edge function which determines an edge (u, v) and its sign probabilities $P(u, v, +)$ and $P(u, v, -)$ (line 4). We first explain how the function determines the edge (u, v). Intuitively, this function divides the whole region of $2^L \times 2^L$ adjacency matrix represented by $\tilde{\mathcal{P}}^{(L)} + \tilde{\mathcal{M}}^{(L)}$ of $\tilde{\mathbf{T}}^{(L)}$ into four quadrants. Then, it selects one of them with the corresponding probability, and repeats the process recursively in the chosen quadrant until the quadrant becomes a single cell where an edge is inserted.

To formalize this process, we need to define *selected region* at level l of Generate-Edge as follows:

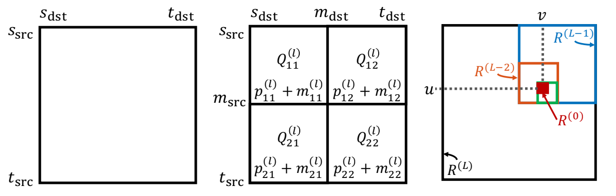
Definition 3.5 (Selected Region). $R^{(l)} = \{[s_{\text{src}}, t_{\text{src}}], [s_{\text{dst}}, t_{\text{dst}}]\}$ represents a region of an adjacency matrix, which is selected at level l, where $[s_{\text{src}}, t_{\text{src}}]$ is a range of source nodes, and $[s_{\text{dst}}, t_{\text{dst}}]$ is that of destination nodes as shown in Figure 8(a). $\blacksquare$

Suppose Generate-Edge is given $R^{(l)}$ at level l. It splits the region $R^{(l)}$ equally into four quadrants $Q_{ij}^{(l)}$ (line 9) for $1 \leq i, j \leq 2$ which are defined as follows:

Algorithm 3: BalanSiNG

Input: seed tensor $\mathrm{T}_{\text{seed}} \in \mathbb{R}^{2\times2\times2}$, target recursion level L, target number $|\mathrm{E}|$ of edges, noise parameter γ, and weight parameter α

1: generate random noises $\mu^{(l)} \in [-\gamma, \gamma]$ [37], and obtain noisy seed tensors $\mathrm{N}^{(l)}_{\text{seed}}$ using Equation (6) with T_{seed} and $\mu^{(l)}$ for $1 \le l \le L$
2: **parallel for** $k \leftarrow 1$ to $|\mathrm{E}|$ **do**
3: set $R^{(L)} \leftarrow \{[1, 2^L], [1, 2^L]\}$ as an initial region
4: $\{+P(u,v,+), -P(u,v,-)\}$ and $(u, v) \leftarrow$ Generate-Edge(L, $R^{(L)}$)
5: compute $P(+|u, v)$ using Equation (2)
6: toss a biased coin with $P(+|u, v)$
if head appears, **then** $\hat{s} \leftarrow +$, **otherwise**, $\hat{s} \leftarrow -$
7: **write** $(u \rightarrow v, \hat{s})$
8: **procedure** Generate-Edge(level l, region $R^{(l)}$)
9: divide $R^{(l)}$ into four quadrants $Q^{(l)}_{ij}$ for $1 \le i, j \le 2$
10: randomly select a quadrant $Q^{(l)}_{ij}$ according to probabilities $p^{(l)}_{ij} + m^{(l)}_{ij}$ in $\mathrm{N}^{(l)}_{\text{seed}}$ for $1 \le i, j \le 2$
11: set $R^{(l-1)} \leftarrow Q^{(l)}_{ij}$ as a selected region for level $l-1$
12: **if** l is 1 **then**
13: **return** $\{+p^{(1)}_{ij}, -m^{(1)}_{ij}\}$ and (u, v) in $R^{(0)}$
14: **else**
15: $\{+\tilde{p}^{(l-1)}_{uv}, -\tilde{m}^{(l-1)}_{uv}\}$ and $(u, v) \leftarrow$ Generate-Edge($l-1$, $R^{(l-1)}$)
16: compute $\{+\tilde{p}^{(l)}_{uv}, -\tilde{m}^{(l)}_{uv}\}$ using Equation (8)
17: **return** $\{+\tilde{p}^{(l)}_{uv}, -\tilde{m}^{(l)}_{uv}\}$ and (u, v)



(a) Region $R^{(l)}$ (b) Quadrants $Q^{(l)}_{ij}$ in $R^{(l)}$ (c) Base region $R^{(0)}$

Figure 8: The concept of region and quadrants.

Definition 3.6 (Quadrants in $R^{(l)}$). Given $R^{(l)}$, let $m_{\text{src}} = s_{\text{src}} + \lfloor \frac{t_{\text{src}} - s_{\text{src}}}{2} \rfloor$ and $m_{\text{dst}} = s_{\text{dst}} + \lfloor \frac{t_{\text{dst}} - s_{\text{dst}}}{2} \rfloor$. Each quadrant $Q^{(l)}_{ij}$ is defined as in Figure 8(b) for $1 \le i, j \le 2$. ∎

Then, it randomly selects a quadrant $Q^{(l)}_{ij}$ with the probability $p^{(l)}_{ij} + m^{(l)}_{ij}$ which is based on the noisy seed tensor $\mathrm{N}^{(l)}_{\text{seed}}$ (line 10). Note that $p^{(l)}_{ij} + m^{(l)}_{ij}$ indicates $P(i, j, +) + P(i, j, -) = P(i, j)$ interpreted as the probability of selecting (i, j)-th quadrant in $R^{(l)}$. For the next level $l-1$, it sets $R^{(l-1)}$ to the selected $Q^{(l)}_{ij}$ (line 11). The function recursively repeats this process for $R^{(l-1)}$ (line 15) until l becomes 1 when the selected region $R^{(0)}$ (called *base region*) is a single cell representing the edge (u, v) (line 13) as shown in Figure 8(c), after starting from the initial region $R^{(L)}$ (line 3).

The edge sign probabilities $P(u, v, +)$ and $P(u, v, -)$ are also recursively computed using the following equation (line 16):

$$\{+\tilde{p}^{(l)}_{uv}, -\tilde{m}^{(l)}_{uv}\} \leftarrow f_\alpha\left(f_b\left(\{+p^{(l)}_{ij}, -m^{(l)}_{ij}\} \otimes \{+\tilde{p}^{(l-1)}_{uv}, -\tilde{m}^{(l-1)}_{uv}\}\right)\right) \quad (8)$$

which is the entry-wise version of Equation (7) where $p^{(l)}_{ij}$ and $m^{(l)}_{ij}$ are the selected quadrant probabilities at line 10 (derivation in Lemma C.1 of Appendix C). The terms $\tilde{p}^{(l)}_{uv}$ and $\tilde{m}^{(l)}_{uv}$ denote the entries of $\tilde{\mathcal{P}}^{(l)}$ and $\tilde{\mathcal{M}}^{(l)}$ corresponding to edge (u, v), respectively, where $\tilde{\mathrm{T}}^{(l)} = \{+\tilde{\mathcal{P}}^{(l)}, -\tilde{\mathcal{M}}^{(l)}\}$. Note that each probability is scalar, i.e., $p^{(l)}_{ij}, m^{(l)}_{ij} \in \mathbb{R}^{1\times1}$; thus, $\{+p^{(l)}_{ij}, -m^{(l)}_{ij}\} \in \mathbb{R}^{1\times1\times2}$. Similarly, $\{+\tilde{p}^{(l-1)}_{uv}, -\tilde{m}^{(l-1)}_{uv}\} \in \mathbb{R}^{1\times1\times2}$. The Kronecker product result in $f_b(\cdot)$ is $\{+p^{(l)}_{ij}\tilde{p}^{(l-1)}_{uv}, -p^{(l)}_{ij}\tilde{m}^{(l-1)}_{uv}, -m^{(l)}_{ij}\tilde{p}^{(l-1)}_{uv}, +m^{(l)}_{ij}\tilde{m}^{(l-1)}_{uv}\} \in \mathbb{R}^{1\times1\times4}$ which is consistent with the input definition of $f_b(\cdot)$ when $N = 1$ (see Definition 3.2). For level $l-1$, $\{+\tilde{p}^{(l-1)}_{uv}, -\tilde{m}^{(l-1)}_{uv}\}$ is recursively computed by Generate-Edge (line 15). The final $\{+\tilde{p}^{(L)}_{uv}, -\tilde{m}^{(L)}_{uv}\}$ returned by the function is $\{+P(u,v,+), -P(u,v,-)\}$ (line 4).

Table 3: BalanSiNG has the smallest time and space complexities. $|\mathrm{E}|$ and $|\mathrm{V}|$ are the number of edges and nodes, respectively, and $d_{\max}$ is the maximum node degree.

Method	Time	Space	Parallel?
IB [45]	$O(\lvert\mathrm{E}\rvert\lvert\mathrm{V}\rvert)$	$O(\lvert\mathrm{E}\rvert)$	No
Evo [29]	$O(d^3_{\max}\lvert\mathrm{E}\rvert\lvert\mathrm{V}\rvert)$	$O(\lvert\mathrm{E}\rvert)$	No
BSCL [6]	$O(d^2_{\max}\lvert\mathrm{E}\rvert + \lvert\mathrm{V}\rvert)$	$O(\lvert\mathrm{E}\rvert)$	No
BalanSiNG (proposed)	$O(\lvert\mathrm{E}\rvert \log \lvert\mathrm{V}\rvert)$	$O(\log \lvert\mathrm{V}\rvert)$	**Yes**

Note that the generation of a signed edge of Generate-Edge is independent of the generation of other edges; thus, Algorithm 3 of BalanSiNG generates signed edges in parallel (line 2). We let Algorithm 3 call the Generate-Edge function in parallel using Apache Spark, a widely used distributed computing framework.

3.4.1 Complexity Analysis. We analyze the complexities of BalanSiNG. To compare BalanSiNG with other sequential methods, we analyze the sequential complexities as follows:

Lemma 3.7 (Complexity of BalanSiNG). *The time complexity of BalanSiNG is $O(|\mathrm{E}| \log |\mathrm{V}|)$ where $|\mathrm{E}|$ and $|\mathrm{V}|$ are the number of edges and nodes, respectively. The space complexity is $O(\log |\mathrm{V}|)$.*

Proof. Let $T(L)$ be the time complexity of Generate-Edge given L; then, $T(L) = T(L-1) + O(1)$ since there is a recursive call with $L-1$ at line 15 of Algorithm 3, and other lines demand $O(1)$. Hence, it is obvious that $T(L)$ is in $O(L) = O(\log |\mathrm{V}|)$ where we set $|\mathrm{V}| = 2^L$. BalanSiNG generates $|\mathrm{E}|$ edges; thus, the total time complexity is $O(|\mathrm{E}| \log |\mathrm{V}|)$. BalanSiNG needs to have L noisy seed tensors $\mathrm{N}^{(l)}_{\text{seed}} \in \mathbb{R}^{2\times2\times2}$ where each tensor exhibits constant space complexity, i.e., $O(1)$ (line 1 in Algorithm 3). Therefore, the space complexity is $O(L) = O(\log |\mathrm{V}|)$. □

Table 3 compares signed network generation methods (see Section 4.1.2) in terms of complexities and parallelism. The time and space complexities of BalanSiNG are less than those of other sequential methods such as IB, Evo, and BSCL. Especially, these competitors require to store all generated edges in memory (i.e., they require $O(|\mathrm{E}|)$ space) since they need to retrieve the common neighbors of two nodes to determine the edge's sign between the nodes based on balance theory. On the other hand, BalanSiNG is free of such restriction; i.e., as soon as an edge is created, BalanSiNG is able to write it onto disk (line 7 of Algorithm 3).

4 EXPERIMENT

We aim to answer the following questions from experiments:

- **Q1. Properties of signed networks (Section 4.2).** Is our proposed BalanSiNG able to synthetically generate signed networks following the desired properties of real-world networks?
- **Q2. Fine-grained comparison of signed triangles (Section 4.3).** Does BalanSiNG generate graphs with realistic signed triangle distributions, compared to other methods?
- **Q3. Effects of parameters (Section 4.4).** How do weight parameter α and recursion level L of BalanSiNG affect the properties of generated networks?
- **Q4. Computational performance (Section 4.5).** How efficient is BalanSiNG for generating large-scale signed networks compared to other competitors? How does BalanSiNG scale up in terms of the number of workers and the data size on distributed machines?

4.1 Experimental Settings

We explain the detailed settings for our experiments.

4.1.1 Datasets. The datasets used for our experiments are summarized in Table 4. The BitcoinO and BitcoinA datasets [22] were extracted from online trust and directed networks served by Bitcoin Alpha and Bitcoin OTC, respectively. The Epinions dataset [10] is a directed signed network, and was scraped from Epinions, a product review site where users are able to mark their trust or distrust to others. We use the datasets to investigate their distinct properties and provide baseline statistics on signed triangle distributions in Table 5.

4.1.2 Competitors. We compare our proposed method BalanSiNG to the following competitors:

- **IB** [45]: IB (Interaction-based model) generates signed edges based on global and local interactions between nodes under ant pheromone mechanism and balance theory.
- **Evo** [29]: Evo (Evolutionary model) randomly generates signed edges, and keeps track of the number of unbalanced triangles over time. Once a node reaches a certain threshold of unbalanced triangle ratio, it randomly removes a link from the node until the threshold is not exceeded.
- **BSCL** [6]: BSCL (Balanced Signed Chung-Lu) is the-state-of-the-art model based on Transitive Chung-Lu model [35] and balance theory, which synthetically produces a signed network by imitating an input signed network.

4.1.3 Parameters. We describe the setting of the parameters for each method as follows:

- **BalanSiNG**: For the weight parameter α, we search for α on a grid between 0 and 1 by 0.05, and choose α which minimizes the absolute difference for edge signs in Equation (12) between a generated network and a real network. We set the noise parameter γ to 0.1 and the seed tensor $\mathbf{T}_{\text{seed}} = \{+\mathcal{P}_{\text{seed}}, -\mathcal{M}_{\text{seed}}\}$ to the following values:
$$\mathbf{T}_{\text{seed}} = \left\{+\begin{bmatrix}0.57 & 0\\ 0 & 0.05\end{bmatrix}, -\begin{bmatrix}0 & 0.19\\ 0.19 & 0\end{bmatrix}\right\} \quad (9)$$
which are derived from $\mathcal{A}_{\text{seed}} = \begin{bmatrix}0.57 & 0.19\\ 0.19 & 0.05\end{bmatrix}$. Many researches [30, 31, 37, 38] have empirically proved that these values produce monotonically decreasing power-law degree distributions. Note that other values of γ and $\mathbf{T}_{\text{seed}}$ can be used as well.
- **IB**: M_G and M_L are the numbers of edges added globally and locally, respectively. p_G and p_L are the probabilities of the positive sign of the globally and locally added edges, respectively. δ is the initial weight of an added edge. ϵ is the parameter for the evaporation. According to their work [45], we set $M_G = M_L$ to 1 and $p_G = p_L$ to $\rho(+)$ for each dataset in Table 4. For δ and ϵ, we perform grid searches from 0 to 1.0 by 0.05 to minimize the absolute difference for edge signs in Equation (12).
- **Evo**: In Evo, α is a friendliness index affecting the formation of the positive sign of an edge, and β is a tolerance threshold for unbalanced triangles. For α and β, we perform grid searches from -1 to 1 by 0.05 to minimize the absolute difference for edge signs in Equation (12).
- **BSCL**: ρ_{BSCL} is a parameter for closing wedge, α_{BSCL} is for creating positive edge, and β_{BSCL} is for closing balanced triangle. Given a real network, those parameters are approximately tuned by the estimation phase of BSCL.

Table 4: Dataset statistics. |V| is the number of nodes, |E| is the number of edges, and $\rho(+)$ is the ratio of positive edges.

Dataset	|V|	|E|	$\rho(+)$	Description
Epinions[1]	131,828	841,372	0.85	Online social network
BitcoinO[2]	5,881	35,592	0.89	Bitcoin social network
BitcoinA[2]	3,783	24,186	0.93	Bitcoin social network
Congress[3]	219	764	0.78	Politician network

[1] http://www.trustlet.org/wiki/Extended_Epinions_dataset
[2] https://snap.stanford.edu/data/soc-sign-bitcoin-otc.html
[3] http://www.cs.cornell.edu/home/llee/data/convote.html

4.1.4 Machines and Implementation. We describe the settings of machines and implementation used for evaluating the computational performance of each method in Section 4.5 as follows:

- **Setting on single machine (Section 4.5.1).** We use a single thread in a machine with an Intel Xeon E3-1240v5 CPU and 32GB RAM, and implement all tested methods including BalanSiNG based on g++ v5.4.0.
- **Setting on distributed machines (Section 4.5.2).** We implement BalanSiNG on Spark to test the scalability on a cluster (managed by Hadoop YARN) that consists of 17 machines: a master and 16 worker nodes. Each worker node has 4 physical cores (Intel Xeon E3-1240v5 CPU) with 32GB RAM, and can run 4 workers. Java v1.8.0, Scala v2.11.8, Hadoop v2.7.3, and Spark v2.11.9 are used.

4.2 Properties of Signed Networks (Q1)

We compare real-world signed network BitcoinO with those generated by BalanSiNG and competitors in Figure 1 to investigate if they exhibit the desired properties of real-world signed networks listed in Section 2.1. We omit the comparisons for other datasets due to the space limit, but the overall tendency is similar. We adjust the parameters of each method so that the resulting networks have almost the same positive edge sign ratios as that of BitcoinO (details in Appendix F); thus, the sign distributions in Figure 1(a) are similar for all graphs.

The signed network generated by BalanSiNG follows the desired properties w.r.t. signs (D1-3) as well as those regardless of signs (D4-6). The balanced triangle distribution is highly skewed as shown in Figure 1(b), and degree distributions follow a power-law as seen from Figure 1(c) to Figure 1(f). The hop plot of BalanSiNG in Figure 1(g) is similar to that of BitcoinO. Also, top-k singular values of graphs from BalanSiNG and BitcoinO monotonically decrease as shown in Figure 1(h).

On the other hand, the signed networks generated by IB and Evo do not follow power-law degree distributions as shown in the third and forth rows (Figure 1(c) to Figure 1(f)). The main reason is that IB and Evo naively create random edges without the consideration of power-law degree distribution. The hop plot and singular value distributions of both methods are also different from those of the real-world network as shown in Figures 1(g) and 1(h). BSCL generates signed networks obeying most of the desired properties, but its balanced triangle distribution (D2) does not; it is not skewed enough compared to the real network (at the first row) and BalanSiNG (at the second row) as shown in Figure 1(b). We further provide the fine-grained comparison about these signed triangles in Section 4.3.

4.3 Fine-grained Comparison of Signed Triangles (Q2)

We compare BalanSiNG to other competitors in terms of signed triangle distribution. As described in Section 2.1, the balanced

Table 5: Comparison of signed triangle distributions by BalanSiNG and competitors. $\rho(\triangle_b)$ and $\rho(\triangle_u)$ indicate the ratios of balanced and unbalanced triangles, respectively. $\rho(\triangle_{+++})$, $\rho(\triangle_{+--})$, $\rho(\triangle_{++-})$, and $\rho(\triangle_{---})$ denote the ratios of the triangle types $\triangle_{+++}$, $\triangle_{+--}$, $\triangle_{++-}$, and $\triangle_{---}$, respectively. Note that BalanSiNG (marked †) generates the most closest signed networks to the corresponding real-world signed networks in terms of absolute difference and Kolmogorov–Smirnov statistic (the lower the better).

Datasets	BitcoinA					BitcoinO					Epinions				
Methods	Real	BalanSiNG†	IB	Evo	BSCL	Real	BalanSiNG†	IB	Evo	BSCL	Real	BalanSiNG†	IB	Evo	BSCL
$\rho(\triangle_b)$	0.8805	0.8740	0.7604	0.8184	0.8366	0.8934	0.8254	0.7023	0.7402	0.7579	0.9240	0.8109	0.7061	0.7025	0.7104
$\rho(\triangle_u)$	0.1195	0.1260	0.2396	0.1816	0.1634	0.1066	0.1746	0.2977	0.2598	0.2422	0.0760	0.1891	0.2939	0.2975	0.2896
Abs. Diff.	-	**0.0130**	0.2402	0.1242	0.0879	-	**0.1360**	0.3822	0.3064	0.2711	-	**0.2261**	0.4358	0.4430	0.4272
K-S Stat.	-	**0.0065**	0.1201	0.0621	0.0439	-	**0.0680**	0.1911	0.1532	0.1356	-	**0.1131**	0.2179	0.2215	0.2136
$\rho(\triangle_{+++})$	0.8413	0.8632	0.7285	0.7954	0.8240	0.8260	0.8014	0.6649	0.6975	0.7281	0.8723	0.7782	0.6688	0.6297	0.6677
$\rho(\triangle_{+--})$	0.0393	0.0108	0.0319	0.0231	0.0126	0.0675	0.0240	0.0374	0.0427	0.0298	0.0517	0.0328	0.0373	0.0728	0.0427
$\rho(\triangle_{++-})$	0.1166	0.1259	0.2377	0.1816	0.1630	0.1026	0.1743	0.2945	0.2598	0.2408	0.0694	0.1886	0.2913	0.2975	0.2875
$\rho(\triangle_{---})$	0.0028	0.0001	0.0009	0.0000	0.0005	0.0040	0.0003	0.0032	0.0000	0.0014	0.0066	0.0005	0.0026	0.0000	0.0021
Abs. Diff.	-	**0.0625**	0.2432	0.1299	0.0927	-	**0.1434**	0.3839	0.3145	0.2764	-	**0.2383**	0.4438	0.4984	0.4362
K-S Stat.	-	**0.0219**	0.1202	0.0621	0.0441	-	**0.0681**	0.1912	0.1533	0.1357	-	**0.1131**	0.2179	0.2426	0.2136

triangle proportion is the most distinct property in real-world signed networks. Hence, we analyze signed triangles of generated signed networks to check if they exhibit distributions similar to that of a real-world signed network.

For the purpose, we first enumerate directed signed triangles in each real signed network as in [43] since all of the signed networks used in this paper are directed, and then measure the ratio $\rho(\cdot)$ for each triangle type. For example, $\triangle_{+++}$ indicates triangles with three positive signs; thus, $\rho(\triangle_{+++}) = |\triangle_{+++}|/|\triangle_{\text{total}}|$ where $|\triangle_{\text{total}}|$ is the total number of triangles. For large-scale signed networks, distributed algorithms [33] can be used to enumerate triangles. Then, we generate synthetic signed networks for each dataset following the procedure in Section 4.2 (see the parameters of each method in Appendix F), and compare the triangle distributions of both real and synthetic networks. To measure the distance between two distributions, we utilize *Absolute Difference* [6] and *Kolmogorov–Smirnov statistic* (K-S statistic) metrics. The absolute difference [6] is defined as the sum of absolute differences between each ratio of real and synthetic triangles (see Definition D.2). The K-S statistic is defined as the maximum gap between the two cumulative distributions; it has been traditionally used for measuring the difference between two distributions. We repeat the above procedure 10 times, and report the average for each method and each dataset. For both of the metrics, small values indicate that the synthetic network has a similar tendency to the corresponding real network in terms of signed triangles.

Table 5 shows the fine-grained comparison on the four types of signed triangles by BalanSiNG and competitors for each dataset. Note that BalanSiNG gives the best signed triangle distribution, showing the smallest absolute difference and K-S statistic. Specifically, BalanSiNG shows about $1.5 \sim 2\times$ better performance than the second best method for each dataset.

4.4 Effect of Parameters (Q3)

We investigate the effect of parameters of BalanSiNG. We focus on the effects of weight parameter α and target recursion level L while noise parameter γ is set to 0.1 and seed ten sor $\mathbf{T}_{\text{seed}}$ is set to the values of Equation (9), as described in Section 4.1. The weight parameter α is introduced to increase the probability of generating positive edges, and the level L controls the size of networks to be generated.

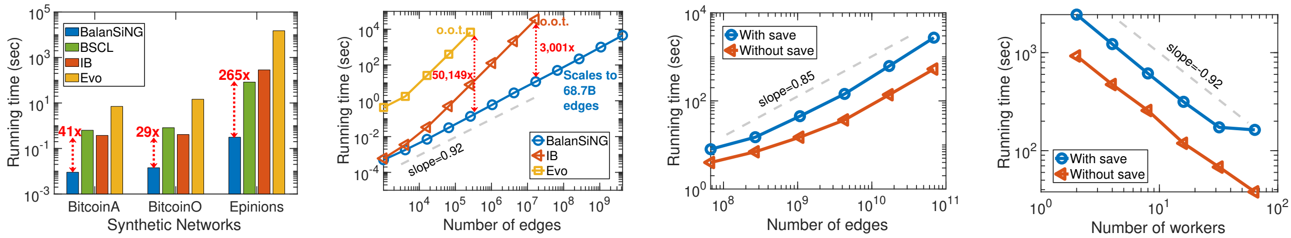
Figure 10(a) shows the effect of the weight parameter α on positive sign ratio $\rho(+)$ and balanced triangle ratio $\rho(\triangle_b)$. We set $L = 17$ and $|\text{E}| = 2^{19}$, and vary α form 0.1 to 0.9. As shown in the figure, both of the ratios $\rho(+)$ and $\rho(\triangle_b)$ increase as α increases. The reason is that according to Definition 3.4, as we increase α, the probability of the positive term becomes large while that of the negative term diminishes. Also, as the number of positive edges increases, balanced triangles $\triangle_{+++}$ and $\triangle_{++-}$ are more likely to be formed. Note that α between 0.7 and 0.85 introduces the skewness of both ratios similarly to those of real signed networks. Thus, our method is able to control the skewness of those ratios according to users' preference through adjusting α.

Figures 10(b) and 10(c) demonstrate the effect of the recursion level L on the ratios $\rho(+)$ and $\rho(\triangle_b)$, and the out-degree distributions of networks generated by BalanSiNG. We set α to 0.8, and vary L from 12 to 21 to generate networks with $|\text{V}| = 2^{L+1}$ and $|\text{E}| = 2^{L+6}$. In Figure 10(b), $\rho(+)$ and $\rho(\triangle_b)$ do not change much as L increases. Also, as shown in Figure 10(c), the degree distributions for different L have almost the same tendency w.r.t. power-law distribution. These results indicate the effect of L on those ratios and degree distribution is marginal, i.e., our method is able to control the size of signed networks to be generated while the tendency of such properties is preserved.

4.5 Computational Performance (Q4)

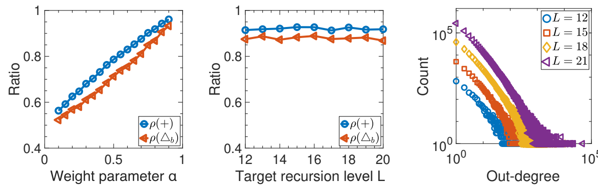
We evaluate the computational performance of BalanSiNG on single and distributed machines.

4.5.1 Performance on Single Machine. We examine the performance of BalanSiNG and competitors on a single machine. The detailed setting is in Section 4.1.4. We fix the size of each synthetic network to that of the corresponding real-world network, and compare the generation time of each method. As shown in Figure 9(a), the generation time of BalanSiNG is up to 265× faster than that of BSCL. Figure 9(b) shows the data scalability of methods. Note that BSCL is excluded since it cannot generate synthetic networks having arbitrary numbers of nodes and edges. BSCL aims to imitate an input network, and thus the size of the generated network of BSCL is fixed to that of the input network. We vary $|V| = 2^{L+1}$ and $|E| = 2^{L+6}$ for $L = 4..26$ where L is the target recursion level. We report out of time (o.o.t.) error when the generation time is more than 24 hours. As shown in Figure 9(b), only BalanSiNG generates the largest network for $L = 26$ within the limited time while IB and Evo generate o.o.t. errors. BalanSiNG is $50,149\times$ and $3,001\times$ faster than Evo and IV, respectively. Furthermore, the slope of BalanSiNG is 0.92, indicating the data scalability of BalanSiNG is near linear w.r.t. the number of edges. To sum, BalanSiNG provides the fastest running time and the best scalability.



(a) Generation time on a single machine (b) Data scalability on a single machine (c) Data scalability on distributed machines (d) Distributed machine scalability

Figure 9: Computational performance of BalanSiNG. (a) BalanSiNG generates signed networks up to 265× **faster than existing methods. (b-c) The data scalability of BalanSiNG is near linear w.r.t. the number of edges on both single and distributed machines where o.o.t. stands for out of time (more than 24 hours). (d) BalanSiNG also scales up well with the increase of the number of workers on distributed machines.**



(a) Effect of weight parameter α (b) Effect of level L on $\rho(+)$ and $\rho(\triangle_b)$ (c) Effect of level L on out-degree distributions

Figure 10: Effects of weight parameter α, and target recursion level L. BalanSiNG generates graphs of various sizes following the power laws, while controlling the positive sign ratio $\rho(+)$ and the balanced triangle ratio $\rho(\triangle_b)$.

4.5.2 Performance on Distributed Machines. We demonstrate the performance of BalanSiNG on distributed machines. The detailed setting is in Section 4.1.4. We report the generation times *with* and *without* writing edges onto disks (line 7 of Algorithm 3). The former is execution time with disk I/O, and the latter is only CPU execution time without disk I/O. To evaluate data scalability, we use 64 workers, and vary $|V| = 2^{L+1}$ and $|E| = 2^{L+6}$ for $L = 20..30$ where L is the target level. Figure 9(c) shows that BalanSiNG has near linear scalability w.r.t. the number of edges with the slope 0.85 in the plot. Note that BalanSiNG generates $|E| = 2^{36} \simeq 68.7$ billion signed edges within 45.5 minutes including disk I/O time on the distributed machines; the generated network is $81,675\times$ larger than the Epinions dataset, the largest real signed network currently open to the public, with respect to the number of edges. Figure 9(d) shows BalanSiNG also scales up well with the increase of the number of workers from 2 to 64 where we set $|V| = 2^{27}$ and $|E| = 2^{32}$. The last point of the blue line at 64 is due to the bottleneck of HDFS I/O, i.e., there are too many workers trying to write edges to disks at the same time.

5 RELATED WORK

Models for generating graphs from scratch. There are various methods for generating unsigned networks following real-world properties described in Section 2.1.2. Barabási et al. [1] proposed Barabási-Albert model through a preferential attachment process for generating scale-free networks. Leskovec et al. [27] identified densification laws and shrinking diameters inherent in graphs over time, and developed Forest Fire for modeling such graphs. Also, they proposed Stochastic Kronecker Graph (SKG) [23], a general generation model that simulates a self-similarity using Kronecker product. They developed FastKronecker [24] that chooses edges in a recursive way to reduce the generation time. However, those models cannot generate signed networks, while BalanSiNG generates signed networks following real-world properties. There are a few methods for generating signed networks from scratch. Vukašinović et al. [45] proposed an interaction based model (IB) using ant pheromone mechanism and balance theory for simulating signed edge generation. Ludwig et al. [29] suggested an evolutionary model (Evo) that simulates an evolving network by inserting or removing signed edges so that the network keeps obeying balance theory. However, their resulting networks give different properties from those of the real-world signed networks, while BalanSiNG generates realistic signed networks as shown in Figure 1.

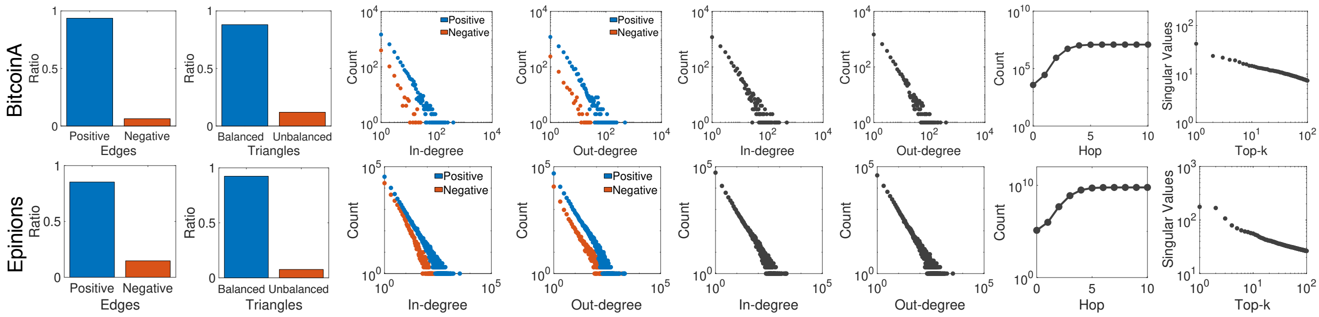
Models for generating graphs imitating an input network. Chung-Lu [35] model aims to generate a synthetic unsigned network by randomly selecting an edge with its associated degree probability. Transitive Chung-LU (TCL) model [35] stochastically performs a two-hop random walk from a node in order to explicitly form at least one triangle, thereby imitating clustering coefficients in the input graph. Derr et al. [6] proposed Balanced Signed Chung-LU (BSCL) model, the state-of-the-art model for synthetic signed networks. They combined balance theory and TCL model in order that the resulting network imitates the signed triangle distribution of the input graph. However, BSCL is not fast, does not generate networks which fully follow the properties of real-world signed networks, and cannot generate signed networks having an arbitrary number of nodes from scratch. On the other hand, BalanSiNG is fast and scalable, generates the most similar networks to real signed networks as in Table 5, and generates graphs of arbitrary sizes as in Figure 9.

6 CONCLUSION

We propose BalanSiNG, a novel, scalable, and fully parallelizable method for generating realistic signed networks from scratch. BalanSiNG exploits the self-similar balanced structure with Kronecker product, and produces realistic signed networks by introducing noises and weights. We implement BalanSiNG in parallel using Spark, a widely used distributed computing platform. Experiments show that BalanSiNG generates the most realistic signed networks. BalanSiNG is up to 265× faster than existing methods for generating signed networks, and scales up near linearly with the size of networks and the number of workers on both single and distributed machines, successfully generating graphs with 68.7 billion edges.

ACKNOWLEDGMENTS

This work was supported by Institute of Information & Communications Technology Planning & Evaluation(IITP) grant funded by the Korea government(MSIT) [2013-0-00179, Development of Core Technology for Context-aware Deep-Symbolic Hybrid Learning and Construction of Language Resources]. The Institute of Engineering Research at Seoul National University provided research facilities for this work. The ICT at Seoul National University provides research facilities for this study. U Kang is the corresponding author.



(a) Global sign proportion (b) Balanced triangle distribution (c) Signed in-degree distribution (d) Signed out-degree distribution (e) In-degree distribution (f) Out-degree distribution (g) Hop plot (h) Singular value distribution

Figure 11: Various properties of real-world signed networks. (a)-(d) illustrate properties derived from edge signs, and (e)-(h) depict traditional properties of real-world networks regardless of edge signs (see Section 2.1).

REFERENCES

[1] Réka Albert and Albert-László Barabási. 2002. Statistical mechanics of complex networks. *Rev. Mod. Phys.* 74 (Jan 2002), 47–97. Issue 1.
[2] Dorwin Cartwright and Frank Harary. 1956. Structural balance: a generalization of Heider's theory. *Psychological review* 63, 5 (1956), 277.
[3] Deepayan Chakrabarti and Christos Faloutsos. 2006. Graph mining: Laws, generators, and algorithms. *ACM computing surveys* 38, 1 (2006), 2.
[4] Deepayan Chakrabarti, Yiping Zhan, and Christos Faloutsos. 2004. R-MAT: A recursive model for graph mining. In *SDM*. SIAM.
[5] Lingyang Chu, Zhefeng Wang, Jian Pei, Jiannan Wang, Zijin Zhao, and Enhong Chen. 2016. Finding gangs in war from signed networks. In *KDD*. ACM.
[6] Tyler Derr, Charu Aggarwal, and Jiliang Tang. 2018. Signed Network Modeling Based on Structural Balance Theory. In *CIKM*. ACM.
[7] David Easley, Jon Kleinberg, et al. 2012. Networks, crowds, and markets: Reasoning about a highly connected world. *Significance* 9 (2012), 43–44.
[8] Victor M Eguiluz, Dante R Chialvo, Guillermo A Cecchi, Marwan Baliki, and A Vania Apkarian. 2005. Scale-free brain functional networks. *Physical review letters* 94, 1 (2005), 018102.
[9] Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos. 1999. On power-law relationships of the internet topology. In *SIGCOMM*, Vol. 29. ACM.
[10] Ramanthan Guha, Ravi Kumar, Prabhakar Raghavan, and Andrew Tomkins. 2004. Propagation of trust and distrust. In *WWW*. ACM.
[11] Paul W Holland and Samuel Leinhardt. 1971. Transitivity in structural models of small groups. *Comparative group studies* 2, 2 (1971), 107–124.
[12] ByungSoo Jeon, Inah Jeon, and U Kang. 2015. TeGViz: Distributed Tera-Scale Graph Generation and Visualization. In *IEEE International Conference on Data Mining Workshop, ICDMW 2015, Atlantic City, NJ, USA, November 14-17, 2015.*
[13] ByungSoo Jeon, JungWoo Lee, and U Kang. 2016. TeT: Distributed Tera-Scale Tensor Generator. *Journal of KIISE* 43, 8 (2016), 910–918.
[14] Woojeong Jin, Jinhong Jung, and U Kang. 2019. Supervised and extended restart in random walks for ranking and link prediction in networks. *PLOS ONE* 14, 3 (03 2019), 1–23.
[15] Jinhong Jung, Woojeong Jin, and U Kang. 2019. Random walk-based ranking in signed social networks: model and algorithms. *Knowledge and Information Systems* (2019).
[16] Jinhong Jung, Woojeong Jin, Lee Sael, and U Kang. 2016. Personalized Ranking in Signed Networks Using Signed Random Walk with Restart.. In *ICDM*. IEEE.
[17] Jinhong Jung, Namyong Park, Lee Sael, and U Kang. 2017. BePI: Fast and Memory-Efficient Method for Billion-Scale Random Walk with Restart. In *SIGMOD*. ACM.
[18] Jinhong Jung, Kijung Shin, Lee Sael, and U Kang. 2016. Random walk with restart on large graphs using block elimination. *ACM Transactions on Database Systems (TODS)* 41, 2 (2016), 12.
[19] U Kang, Charalampos E. Tsourakakis, and Christos Faloutsos. 2009. PEGASUS: A Peta-Scale Graph Mining System. In *ICDM*.
[20] Junghwan Kim, Haekyu Park, Ji-Eun Lee, and U Kang. 2018. SIDE: Representation Learning in Signed Directed Networks. In *WWW*. ACM.
[21] Srijan Kumar, Francesca Spezzano, and VS Subrahmanian. 2014. Accurately detecting trolls in slashdot zoo via decluttering. In *ASONAM*. IEEE.
[22] Srijan Kumar, Francesca Spezzano, VS Subrahmanian, and Christos Faloutsos. 2016. Edge Weight Prediction in Weighted Signed Networks.. In *ICDM*. IEEE.
[23] Jurij Leskovec, Deepayan Chakrabarti, Jon Kleinberg, and Christos Faloutsos. 2005. Realistic, mathematically tractable graph generation and evolution, using kronecker multiplication. In *ECML/PKDD*. Springer.
[24] Jure Leskovec, Deepayan Chakrabarti, Jon M. Kleinberg, Christos Faloutsos, and Zoubin Ghahramani. 2010. Kronecker Graphs: An Approach to Modeling Networks. *JMLR* 11 (2010), 985–1042.
[25] Jure Leskovec, Daniel Huttenlocher, and Jon Kleinberg. 2010. Predicting positive and negative links in online social networks. In *WWW*. ACM.
[26] Jure Leskovec, Daniel Huttenlocher, and Jon Kleinberg. 2010. Signed networks in social media. In *CHI*. ACM.
[27] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. 2005. Graphs over time: densification laws, shrinking diameters and possible explanations. In *KDD*.
[28] Xiaoming Li, Hui Fang, and Jie Zhang. 2019. Supervised User Ranking in Signed Social Networks. In *AAAI*, Vol. 33. 184–191.
[29] Mark Ludwig and Peter Abell. 2007. An evolutionary model of social networks. *The European Physical Journal B* 58, 1 (2007), 97–105.
[30] Richard C Murphy, Kyle B Wheeler, Brian W Barrett, and James A Ang. 2010. Introducing the graph 500. *Cray Users Group (CUG)* 19 (2010), 45–74.
[31] Himchan Park and Min-Soo Kim. 2017. TrillionG: A trillion-scale synthetic graph generator using a recursive vector model. In *SIGMOD*. ACM.
[32] Ha-Myung Park, Namyong Park, Sung-Hyon Myaeng, and U Kang. 2016. Partition Aware Connected Component Computation in Distributed Systems. In *ICDM*.
[33] Ha-Myung Park, Sung-Hyon Myaeng, and U Kang. 2016. Pte: Enumerating trillion triangles on distributed systems. In *KDD*. ACM.
[34] Ha-Myung Park, Chiwan Park, and U Kang. 2018. PegasusN: A Scalable and Versatile Graph Mining System. In *AAAI*.
[35] Joseph J Pfeiffer, Timothy La Fond, Sebastian Moreno, and Jennifer Neville. 2012. Fast generation of large scale social networks while incorporating transitive closures. In *PASSAT*. IEEE, 154–165.
[36] Manfred Schroeder. 2009. *Fractals, chaos, power laws: Minutes from an infinite paradise.* Courier Corporation.
[37] Comandur Seshadhri, Ali Pinar, and Tamara G Kolda. 2011. An in-depth study of stochastic Kronecker graphs. In *ICDM*. IEEE.
[38] Comandur Seshadhri, Ali Pinar, and Tamara G Kolda. 2013. An in-depth analysis of stochastic Kronecker graphs. *J. ACM* 60, 2 (2013), 13.
[39] Kijung Shin, Jinhong Jung, Lee Sael, and U Kang. 2015. BEAR: Block Elimination Approach for Random Walk with Restart on Large Graphs. In *SIGMOD*.
[40] Dongjin Song, David A Meyer, and Dacheng Tao. 2015. Efficient latent link recommendation in signed networks. In *KDD*. ACM.
[41] Michael Szell, Renaud Lambiotte, and Stefan Thurner. 2010. Multirelational organization of large-scale social networks in an online world. *Proceedings of the National Academy of Sciences* 107, 31 (2010), 13636–13641.
[42] Jiliang Tang, Charu Aggarwal, and Huan Liu. 2016. Node classification in signed social networks. In *SDM*. SIAM.
[43] Jiliang Tang, Yi Chang, Charu Aggarwal, and Huan Liu. 2016. A survey of signed network mining in social media. *Comput. Surveys* 49, 3 (2016), 42.
[44] Matt Thomas, Bo Pang, and Lillian Lee. 2006. Get out the vote: Determining support or opposition from Congressional floor-debate transcripts. In *EMNLP*. Association for Computational Linguistics.
[45] Vida Vukašinović, Jurij Šilc, and Risth Škrekovski. 2014. Modeling acquaintance networks based on balance theory. *International Journal of Applied Mathematics and Computer Science* 24, 3 (2014), 683–696.
[46] Suhang Wang, Jiliang Tang, Charu Aggarwal, Yi Chang, and Huan Liu. 2017. Signed network embedding in social media. In *SDM*. SIAM, 327–335.
[47] Pinghua Xu, Wenbin Hu, Jia Wu, and Bo Du. 2019. Link Prediction with Signed Latent Factors in Signed Social Networks. In *SIGKDD*. ACM, 1046–1054.
[48] Bo Yang, William Cheung, and Jiming Liu. 2007. Community mining from signed social networks. *TKDE* 19, 10 (2007), 1333–1348.

APPENDIX

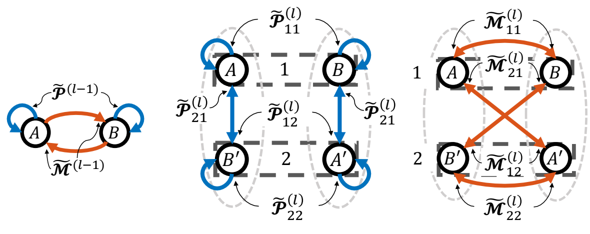
A PROPERTIES OF SIGNED NETWORKS

Figure 11 shows properties of other real-world signed networks. The properties of the BitcoinO dataset are depicted in Figure 1.

B PROOF OF LEMMA 3.3

Proof. We use mathematical induction. For the base case, $\tilde{\mathrm{T}}^{(1)} = \mathrm{T}_{\mathrm{seed}}$ is trivially fully balanced as shown in Figure 5(a). Assume $\tilde{\mathrm{T}}^{(l-1)}$ is fully balanced. Then, $\tilde{\mathrm{T}}^{(l)}$ of Equation (4) with $\mathrm{T}_{\mathrm{seed}} = \tilde{\mathrm{T}}^{(1)}$ is represented as follows:

$$f_b(\tilde{\mathrm{T}}^{(1)} \otimes \tilde{\mathrm{T}}^{(l-1)})$$
$$= \{+(\mathcal{P} \otimes \tilde{\mathcal{P}}^{(l-1)} + \mathcal{M} \otimes \tilde{\mathcal{M}}^{(l-1)}), -(\mathcal{P} \otimes \tilde{\mathcal{M}}^{(l-1)} + \mathcal{M} \otimes \tilde{\mathcal{P}}^{(l-1)})\}$$



(a) $\{+\tilde{\mathcal{P}}^{(l-1)}, -\tilde{\mathcal{M}}^{(l-1)}\}$ (b) $\tilde{\mathcal{P}}^{(l)}$ (c) $\tilde{\mathcal{M}}^{(l)}$

Figure 12: Structures of (a) $\tilde{\mathcal{P}}^{(l-1)}$&$\tilde{\mathcal{M}}^{(l-1)}$, (b) $\tilde{\mathcal{P}}^{(l)}$, & (c) $\tilde{\mathcal{M}}^{(l)}$.

$$= \left\{ +\begin{bmatrix} p_{11}\tilde{\mathcal{P}}^{(l-1)} & m_{12}\tilde{\mathcal{M}}^{(l-1)} \\ m_{21}\tilde{\mathcal{M}}^{(l-1)} & p_{22}\tilde{\mathcal{P}}^{(l-1)}\end{bmatrix}, -\begin{bmatrix} p_{11}\tilde{\mathcal{M}}^{(l-1)} & m_{12}\tilde{\mathcal{P}}^{(l-1)} \\ m_{21}\tilde{\mathcal{P}}^{(l-1)} & p_{22}\tilde{\mathcal{M}}^{(l-1)}\end{bmatrix}\right\}$$

$$= \left\{ +\begin{bmatrix}\tilde{\mathcal{P}}^{(l)}_{11} & \tilde{\mathcal{P}}^{(l)}_{12}\\ \tilde{\mathcal{P}}^{(l)}_{21} & \tilde{\mathcal{P}}^{(l)}_{22}\end{bmatrix}, -\begin{bmatrix}\tilde{\mathcal{M}}^{(l)}_{11} & \tilde{\mathcal{M}}^{(l)}_{12}\\ \tilde{\mathcal{M}}^{(l)}_{21} & \tilde{\mathcal{M}}^{(l)}_{22}\end{bmatrix}\right\} = \left\{+\tilde{\mathcal{P}}^{(l)}, -\tilde{\mathcal{M}}^{(l)}\right\}$$

Figure 12(a) shows $\tilde{\mathrm{T}}^{(l-1)}$ where $\tilde{\mathcal{P}}^{(l-1)}$ represents edges within each group (A and B), and $\tilde{\mathcal{M}}^{(l-1)}$ represents edges between the two groups. We depict the block structure of $\tilde{\mathcal{P}}^{(l)}$ in Figure 12(b) where $\tilde{\mathcal{P}}^{(l)}_{ij}$ indicates (i, j)-th block of $\tilde{\mathcal{P}}^{(l)}$. Figure 12(b) has two copies: (A, B) of 1st copy and (A', B') of 2nd copy. Then, $\tilde{\mathcal{P}}^{(l)}_{11}$ means edges within A and B of 1st copy because they are from $p_{11}\tilde{\mathcal{P}}^{(l-1)} = \tilde{\mathcal{P}}^{(l)}_{11}$. Also, $\tilde{\mathcal{P}}^{(l)}_{12}$ represents directed edges from A to B', and from B to A' by $m_{12}\tilde{\mathcal{M}}^{(l-1)}$. Other blocks in $\tilde{\mathcal{P}}^{(l)}$ are similarly interpreted; thus, there are two groups (A, B') and (A', B) having positive between-group edges in the graph of $\tilde{\mathcal{P}}^{(l)}$. Each block in $\tilde{\mathcal{M}}^{(l)}$ represents edges between the groups as shown in Figure 12(c). These indicate $\tilde{\mathrm{T}}^{(l)}$ is also fully balanced. Hence, $\tilde{\mathrm{T}}^{(l)}$ is fully balanced for any $l \geq 1$. □

C LEMMA OF ENTRY-WISE RECURSIVE REPRESENTATION OF BALANSING

Lemma C.1. *Let $R^{(l)}$ be the selected region at level l with probability $p^{(l)}_{ij} + m^{(l)}_{ij}$ in* Generate-Edge. *Let (u, v) be decided through $R^{(L)}, \cdots, R^{(0)}$. Equation (7) for (u, v) is equivalent to Equation (8).*

Proof. Equation (7) is represented as follows:

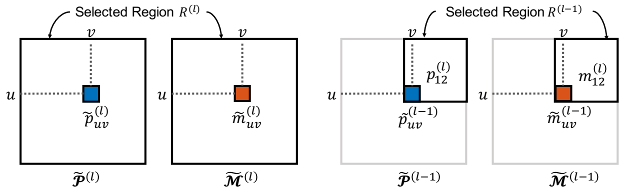
$$\tilde{\mathrm{T}}^{(l)} = f_\alpha(f_b(\mathrm{N}^{(l)}_{\mathrm{seed}} \otimes \tilde{\mathrm{T}}^{(l-1)})) \Leftrightarrow \tag{10}$$
$$\{+\tilde{\mathcal{P}}^{(l)}, -\tilde{\mathcal{M}}^{(l)}\} = f_\alpha(f_b(\{+\mathcal{P}^{(l)}_{\mathrm{seed}}, -\mathcal{M}^{(l)}_{\mathrm{seed}}\} \otimes \{+\tilde{\mathcal{P}}^{(l-1)}, -\tilde{\mathcal{M}}^{(l-1)}\}))$$

Let $\tilde{p}^{(l)}_{uv}$ and $\tilde{m}^{(l)}_{uv}$ indicate the fixed location (u, v) in $\tilde{\mathcal{P}}^{(l)}$ and $\tilde{\mathcal{M}}^{(l)}$ under $R^{(l)}$ as shown in Figure 13(a). Let $g(\cdot)$ be a function that extracts entries participating in the computation related to (u, v) in a signed tensor of Equation (7). For $\{+\tilde{\mathcal{P}}^{(l)}, -\tilde{\mathcal{M}}^{(l)}\}$, $g(\cdot)$ extracts $\tilde{p}^{(l)}_{uv}$ and $\tilde{m}^{(l)}_{uv}$:

$$\{+\tilde{p}^{(l)}_{uv}, -\tilde{m}^{(l)}_{uv}\} \leftarrow g\left(\{+\tilde{\mathcal{P}}^{(l)}, -\tilde{\mathcal{M}}^{(l)}\}, (u, v)\right)$$

Note that $R^{(l-1)}$ is a selected region with probability $p^{(l)}_{ij} + m^{(l)}_{ij}$ where $p^{(l)}_{ij} \in \mathcal{P}^{(l)}_{\mathrm{seed}}$ and $m^{(l)}_{ij} \in \mathcal{M}^{(l)}_{\mathrm{seed}}$. As shown in Figure 13(b), suppose $p^{(l)}_{ij}$ and $m^{(l)}_{ij}$ correspond to $(1, 2)$-th quadrant, respectively, i.e., $p^{(l)}_{ij} = p^{(l)}_{12}$ and $m^{(l)}_{ij} = m^{(l)}_{12}$. Then, other quadrant probabilities except for $p^{(l)}_{12}$ and $m^{(l)}_{12}$ do not affect the computation of $\{+\tilde{p}^{(l)}_{uv}, -\tilde{m}^{(l)}_{uv}\}$ through Kronecker product. Also, since (u, v) is fixed, the only locations corresponding to (u, v) of $\tilde{\mathcal{P}}^{(l-1)}$ and $\tilde{\mathcal{M}}^{(l-1)}$ affect the final result as shown in Figure 13(b). In other words, only $\tilde{p}^{(l-1)}_{uv}$ and $\tilde{m}^{(l-1)}_{uv}$ participate in the computation for $\{+\tilde{p}^{(l)}_{uv}, -\tilde{m}^{(l)}_{uv}\}$, and $\{+\tilde{p}^{(l-1)}_{uv}, -\tilde{m}^{(l-1)}_{uv}\}$ are recursively obtained by $g(\cdot)$ as follows:

$$\{+\tilde{p}^{(l-1)}_{uv}, -\tilde{m}^{(l-1)}_{uv}\} \leftarrow g\left(\{+\tilde{\mathcal{P}}^{(l-1)}, -\tilde{\mathcal{M}}^{(l-1)}\}, (u, v)\right)$$



(a) Recursion level l (b) Recursion level $l-1$

Figure 13: The locations corresponding to (u, v) in (a) $\{+\tilde{\mathcal{P}}^{(l)}, -\tilde{\mathcal{M}}^{(l)}\}$ and (b) $\{+\tilde{\mathcal{P}}^{(l-1)}, -\tilde{\mathcal{M}}^{(l-1)}\}$.

Hence, Equation (10) is represented with $g(\cdot)$ as follows:

$$g\left(\{+\tilde{\mathcal{P}}^{(l)}, -\tilde{\mathcal{M}}^{(l)}\}, (u, v)\right)$$
$$= f_\alpha\left(f_b\left(\{+p^{(l)}_{ij}, -m^{(l)}_{ij}\} \otimes g\left(\{+\tilde{\mathcal{P}}^{(l-1)}, -\tilde{\mathcal{M}}^{(l-1)}\}, (u, v)\right)\right)\right)$$
$$\Leftrightarrow \{+\tilde{p}^{(l)}_{uv}, -\tilde{m}^{(l)}_{uv}\} = f_\alpha(f_b(\{+p^{(l)}_{ij}, -m^{(l)}_{ij}\} \otimes \{+\tilde{p}^{(l-1)}_{uv}, -\tilde{m}^{(l-1)}_{uv}\})). \quad \square$$

Note that Generate-Edge$(\cdot)$ represents the recursive function $g(\cdot)$, and $\tilde{p}^{(L)}_{uv} = P(u, v, +)$ and $\tilde{m}^{(L)}_{uv} = P(u, v, -)$.

D DEFINITIONS

Definition D.1 (Kronecker Product). Given $\mathbf{A} \in \mathbb{R}^{m\times n}$ and $\mathbf{B} \in \mathbb{R}^{p\times q}$, the Kronecker product of $\mathbf{A}$ and $\mathbf{B}$ is defined as follows:

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}\mathbf{B} & \cdots & a_{1n}\mathbf{B} \\ \vdots & \ddots & \vdots \\ a_{m1}\mathbf{B} & \cdots & a_{mn}\mathbf{B}\end{bmatrix}$$

where a_{ij} is the (i, j)-th entry of $\mathbf{A}$, and $\mathbf{A} \otimes \mathbf{B} \in \mathbb{R}^{mp\times nq}$. ■

Definition D.2 (Absolute Difference for Signed Triangles and Edge Signs [6]). Let $\rho_{\mathrm{real}}(\cdot)$ and $\rho_{\mathrm{syn}}(\cdot)$ denote ratios from a real network and a synthetic network, respectively. Let T be the set of signed triangles, i.e., $\mathrm{T} = \{\triangle_{+++}, \triangle_{++-}, \triangle_{+--}, \triangle_{---}\}$. Then, *absolute difference* for signed triangles is defined as follows:

$$\text{Abs. Diff. (T)} = \sum_{\triangle\in\mathrm{T}} |\rho_{\mathrm{real}}(\triangle) - \rho_{\mathrm{syn}}(\triangle)| \tag{11}$$

Let S be the set of signs, i.e., $\mathrm{S} = \{+, -\}$. Then, *absolute difference* for edge signs is defined as follows:

$$\text{Abs. Diff. (S)} = \sum_{s\in\mathrm{S}} |\rho_{\mathrm{real}}(s) - \rho_{\mathrm{syn}}(s)| \tag{12}$$

E CONNECTION TO SKG AND NOISY SKG

In terms of edge determination process (line 7 in Algorithm 1 and line 8 in Algorithm 2) without signs, SKSG-B and SKSG are equivalent to Stochastic Kronecker Graph (SKG) [23] and Noisy SKG [37], respectively. SKG constructs a stochastic adjacency matrix $\mathcal{A}$ using Kronecker product where each entry $\mathcal{A}_{uv}$ indicates a probability $P(u, v)$ of forming edge $u \longrightarrow v$. In our models, the probability $P(u, v)$ is divided into $P(u, v, +)$ and $P(u, v, -)$, i.e., $P(u, v) = P(u, v, +) + P(u, v, -)$, implying that $\mathcal{A} = \mathcal{P} + \mathcal{M}$ where $\{+\mathcal{P}, -\mathcal{M}\}$ is a stochastic signed tensor. Thus, the formation of edges without signs in SKSG-B is equivalent to that of SKG; consequently, networks from SKSG-B naturally inherit characteristics of those of SKG. Similarly, the edge formation of SKSG with noises corresponds to that of Noisy SKG.

F PARAMETER SETTING

Table 6 describes the selected α and the target recursion level L of BalanSiNG for each dataset.

Table 6: Parameters used in BalanSiNG

Parameters	BitcoinA	BitcoinO	Epinions
α	0.84	0.75	0.65
L	12	13	17