GSCALER: Synthetically Scaling A Given Graph

J.W. Zhang
National University of Singapore
jiangwei@nus.edu.sg

Y.C. Tay
National University of Singapore
dcstayyc@nus.edu.sg

ABSTRACT

Enterprises and researchers often have datasets that can be represented as graphs (e.g. social networks). The owner of a large graph may want to scale it down to a smaller version, e.g. for application development. On the other hand, the owner of a small graph may want to scale it up to a larger version, e.g. to test system scalability. This paper investigates the *Graph Scaling Problem* (GSP):

> *Given a directed graph G and positive integers $\tilde{n}$ and $\tilde{m}$, generate a similar directed graph $\tilde{G}$ with $\tilde{n}$ nodes and $\tilde{m}$ edges.*

This paper presents a graph scaling algorithm GSCALER for GSP. Analogous to DNA shotgun sequencing, GSCALER, decomposes G into small pieces, scales them, then uses the scaled pieces to construct $\tilde{G}$. This construction is based on the indegree/outdegree correlation of nodes and edges.

Extensive tests with real graphs show that GSCALER is scalable and, for many graph properties, it generates a $\tilde{G}$ that has greater similarity to G than other state-of-the-art solutions, like Stochastic Kronecker Graph and *UpSizeR*.

1. INTRODUCTION

The emergence of online social networks, like Facebook and Twitter, has attracted considerable research. However, their enormous sizes make any experiment on the entire graph impractical. It is therefore often necessary to obtain a smaller version of the graph for experiments. We call this the **scaling down** problem.

At the other end of the scale, a new social network service provider may have a small graph, but wants to test the scalability of their system. They may therefore want to have a larger (and necessarily) synthetic version of their current empirical graph. We call this the **scaling up** problem.

These two problems arise in other contexts as well, e.g. where the graph represents router topology or web links. They illustrate the *Graph Scaling Problem (GSP)*:

> *Given a directed graph G and positive integers $\tilde{n}$ and $\tilde{m}$, generate a similar directed graph $\tilde{G}$ with $\tilde{n}$ nodes and $\tilde{m}$ edges.*

There are many possible ways to define "similarity", depending on the context, but we believe the definitions must all be in terms of graph properties, like indegree distribution, clustering coefficient, effective diameter, etc.

However, it is impossible for G and $\tilde{G}$ to have exactly the same properties; e.g. if G and $\tilde{G}$ have the same degree distributions, then the larger graph must necessarily have smaller density. One must therefore select the graph properties that are to be preserved when scaling. GSP facilitates this selection by allowing $\tilde{n}$ and $\tilde{m}$ to be specified separately.

Related work in the literature have objectives that are different from GSP. There are many papers on graph sampling, such as gSH, BFS, forest fire and frontier sampling [1, 3, 18, 21, 23, 30, 35]. They can be viewed as examples of scaling down, since they produce a $\tilde{G}$ that is a subgraph of G; this can have data protection issues that do not arise if $\tilde{G}$ is synthetic. Moreover, graph sampling cannot generate a $\tilde{G}$ that is larger than G.

Other related work use generative models that can produce a $\tilde{G}$ that is smaller or larger than G. For example, an Erdös-Rényi model generates a graph of any size n with a specified edge probability p [9]; Chung and Lu's model generates graphs with a specified degree distribution [4]; and Stochastic Kronecker Graphs [19, 20] are generated from an initiator by applying Kronecker product. However, these do not allow a choice of both $\tilde{n}$ and $\tilde{m}$.

In contrast, we propose GSCALER, a solution to GSP that deviates from previous work by using a technique that is analogous to DNA shotgun sequencing [31]. The latter breaks a long DNA strand into smaller ones that are easier to sequence, then use these smaller sequences to reconstruct the sequence in the original strand.

Similarly, GSCALER (i) breaks the given G into two sets S_{in} and S_{out} of small pieces, (ii) scales them by size to $\widetilde{S_{in}}$ and $\widetilde{S_{out}}$; (iii) merges these pieces to give a set $\widetilde{S_{bi}}$ of larger pieces, then (iv) assembles $\tilde{G}$ from the pieces in $\widetilde{S_{bi}}$.

This paper makes the following contributions:

1. We present GSCALER, an algorithm for solving GSP.
2. We prove that GSCALER (i) does not generate multiple edges between two nodes, and (ii) has small degree distribution error even when the average degree of $\tilde{G}$ differs from that of G.

3. We present experiments that compare Gscaler to 4 other techniques, using 2 real graphs and 7 properties.

We begin by surveying related work in Sec. 2. We describe Gscaler in Sec. 3, and prove that it does not generate multiple edges between any two nodes. Sec. 4 reviews the graph properties, state-of-the-art algorithms and datasets that are used for comparison. Sec. 5 then proves that Gscaler has small error, and presents the experimental comparison to other algorithms. We discuss the choice of $\widetilde{n}$ and $\widetilde{m}$ in Sec. 6, before Sec. 7 concludes with a summary.

2. RELATED WORK

The closest work in graph scaling from the literature are graph sampling algorithms and generative models.

For graph sampling, the main approaches are node-based sampling, edge-based sampling and traversal-based sampling which produce a subgraph of the original graph G.

Node-based sampling selects a set of nodes V_{sub} from G, then $\widetilde{G}$ is just the induced graph of this set of nodes V_{sub}. Authors in [32] pointed out that node-based sampling may not preserve a power law degree distribution because of bias induced by high degree nodes.

Similarly, traditional edge-based sampling selects edges randomly. However, this might result in a sparsely connected graph $\widetilde{G}$ [21]. Some other edge-based sampling variants [1,16,21] sample the graph using edge selection/deletion and combination with node selection/deletion.

Most graph sampling techniques focus on traversal-based sampling [14]. Breadth first sampling (BFS) [3,18,35] and random walk sampling (RW) [12, 30] are the most basic and well-known algorithms. Similar to BFS, snow ball sampling [13] (SBS) is widely used in sociology studies. Metropolis-Hastings Random Walk (MHRW) [12,27] is a Markov-Chain Monte Carlo algorithm which samples unbiased subgraph in undirected social graphs. However, MHRW suffers from *sample-rejection problem*. Later, rejection-controlled Metropolis-Hastings (RCMH) [26] is proposed to reduce the sample-rejection ratio.

Frontier sampling is a multi-dimensional random walk which results in better estimators for some graph properties [30]. A probabilistic version of SBS, forest fire (FF) [21,23] captures some important observations in real social networks, e.g. small diameter.

Most traversal-based sampling requires random access to a node's neighbors, which might not be feasible for large graphs (that cannot fit into memory). Hence, streaming graph sampling algorithms are proposed, e.g. induced edge sampling (ES-i) [2]. As mentioned previously, graph sampling algorithms are limited to *scaling down* problem.

For generative models, the Erdös-Rényi model generates a graph of any size n with a specified edge probability p [9]. There are variants of random models that generate graphs with specific graph properties [4, 28], e.g. the Chung-Lu model generates graphs with a specified degree distribution.

One group of generative models [5,6,11,17] employ the strategy of *preferential attachment*. They obey a simple rule: a new node u attaches to the graph at each time step, and adds an edge e_{uv} to an existing node v with a probability p proportional to the degree of the node v.

Another type of generative models is *recursive matrix* model [7,19,20], which recursively multiplies the adjacency matrix. For example, Stochastic Kronecker Graph (SKG) [20] recursively multiplies the graph initiator K_1 through Kronecker product, which results in a self-similar graph. SKG captures most social network properties, such as small diameter and power law degree distribution.

Notation	Description
$G(V,E)$	original graph
$\widetilde{G}(\widetilde{V},\widetilde{E})$	scaled graph
$n/\widetilde{n}$	number of nodes in $G/\widetilde{G}$
$m/\widetilde{m}$	number of edges in $G/\widetilde{G}$
$f_{in}/\widetilde{f_{in}}$	$G/\widetilde{G}$'s indegree distribution
$f_{out}/\widetilde{f_{out}}$	$G/\widetilde{G}$ graph's outdegree distribution
$f_{bi}/\widetilde{f_{bi}}$	$G/\widetilde{G}$ graph's bidegree distribution
$f_{corr}/\widetilde{f_{corr}}$	$G/\widetilde{G}$ graph's edge correlation distribution
$S_{in}/\widetilde{S_{in}}$	set of pieces with incoming edges in $G/\widetilde{G}$
$S_{out}/\widetilde{S_{out}}$	set of pieces with outgoing edges in $G/\widetilde{G}$
$\widetilde{S_{bi}}$	set of pieces with incoming and outgoing edges in $\widetilde{G}$
ct_{Δ}	count function of the pieces in set Δ, Δ can be $S_{in}, \widetilde{S_{in}}$ and so on.
$I(\alpha')/O(\alpha')$	total number of available incoming/outgoing edges for nodes with bidegree α'
$D(f,\widetilde{f})$	KS-D statistics of between f and $\widetilde{f}$

Table 1: Notation

Scaling problem appears in other fields as well. For example, *UpSizeR* is a pioneer tool which synthetically scales a relational dataset [33]. *UpSizeR*'s focus is on preserving correlation among tuples from multiple tables. In relational terms, *GSP* requires preservation of correlation among tuples in a single table (for the edges).

For Resource Description Framework (RDF), the AO benchmark [8] is the first tool that scales down an RDF dataset. Later, RBench [29] is proposed to both scale down and up. However, these two benchmarks are evaluated with different metrics (*dataset coherence, relationship specialty, literal diversity*), so it would be unfair to use them for comparison.

3. GRAPH SCALER (Gscaler)

Given a graph $G(V,E)$, $|V|$ and $|E|$ may need to scale by different factors to maintain similarity for certain properties (e.g. density). Hence, Gscaler allows the user to specify the target $\widetilde{n}$ and $\widetilde{m}$. As shown in Fig.1, the scaling has the following 4 steps:

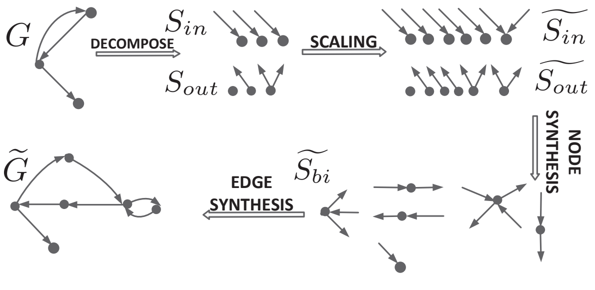


Figure 1: The 4 steps in Gscaler.

Gscaler first decomposes G into 2 sets S_{in} and S_{out}. S_{in} consists of **pieces**, where a piece is a node with its incoming edges (minus the source nodes). Similarly, S_{out} consists of

Algorithm 1: GSCALER(G, $\widetilde{n}$, $\widetilde{m}$)

```
1 $S_{in}$, $S_{out}$, $f_{bi}$, $f_{corr}$ = DECOMPOSE($G$)
2 $\widetilde{S_{in}}$ = SCALING($S_{in}$, $\widetilde{n}$, $\widetilde{m}$)
3 $\widetilde{S_{out}}$ = SCALING($S_{out}$, $\widetilde{n}$, $\widetilde{m}$)
4 $\widetilde{S_{bi}}$ = NODE_SYNTHESIS($\widetilde{S_{in}}$, $\widetilde{S_{out}}$, $f_{bi}$)
5 $\widetilde{G}$ = EDGE_SYNTHESIS($\widetilde{S_{bi}}$, $f_{corr}$)
```

pieces, where a piece is a node with its outgoing edges (minus the target nodes). Each node in G generates 2 pieces, one in S_{in} and one in S_{out}.

After that, GSCALER scales S_{in} and S_{out} to get the scaled sets of pieces $\widetilde{S_{in}}$ and $\widetilde{S_{out}}$ with $\widetilde{n}$ nodes and $\widetilde{m}$ edges.

In node synthesis, $\widetilde{S_{in}}$ and $\widetilde{S_{out}}$ are used to form a set $\widetilde{S_{bi}}$ of larger pieces. Each new piece has a node with incoming edges (with no source nodes) and outgoing edges (with no target nodes); it will generate one node in $\widetilde{G}$.

The last step (edge synthesis) is to link pieces in $\widetilde{S_{bi}}$ which finally results in $\widetilde{G}$. Edge synthesis is similar to a jigsaw puzzle, where you want to fit the small pieces together. Algo. 1 summaries the workflow for GSCALER.

In the following, we will explain each step in detail, and use a running example to show how G is scaled to $\widetilde{G}$ in Fig.1. As presented in Fig.1, $n = 3$, $\tilde{n} = 6$, $m = 3$, $\tilde{m} = 7$.

3.1 DECOMPOSE

This step is straightforward, and Fig. 1 shows the pieces in S_{in} and S_{out} from decomposing G.

3.2 SCALING

To simplify the explanation, we just use S_{in} for demonstration. Let $N = \{0, 1, 2, 3, \ldots\}$. We use the *count* function ct to denote k pieces in Δ has property x:

$$ct_\Delta(x) = k, \;\; k \in N \tag{1}$$

For example, $ct_{S_{in}}(x) = k$ means k pieces in S_{in} have indegree x. For both S_{in} and S_{out}, the scaling process takes the following three steps:

- **node scaling**. For indegree x, let $A_x = ct_{S_{in}}(x) \times \frac{\widetilde{n}}{n}$. As A_x might not be an integer, we round-off $\widetilde{S_{in}}$ as follows:

$$ct_{\widetilde{S_{in}}}(x) = \begin{cases} \lceil A_x \rceil & \text{with probability } A_x - \lfloor A_x \rfloor \\ \lfloor A_x \rfloor & \text{with probability } \lceil A_x \rceil - A_x \end{cases} \tag{2}$$

For example, if $A_x = 4.8$, then with 0.8 probability, $ct_{\widetilde{S_{in}}}(x) = 5$, and with 0.2 probability, $ct_{\widetilde{S_{in}}}(x) = 4$. Hence, Eq.2 can be rewritten as

$$\mathrm{E}[ct_{\widetilde{S_{in}}}(x)] = ct_{S_{in}}(x) \times \frac{\widetilde{n}}{n} \tag{3}$$

Fig.2 shows the $\widetilde{S_{in}}$ and $\widetilde{S_{out}}$ that we get after this scaling.

- **node adjustment**. With randomness in node scaling (Eq.2), we may have $|\widetilde{S_{in}}| \neq \widetilde{n}$. If so, $|\widetilde{n} - |\widetilde{S_{in}}||$ nodes with random degree are added to or removed from $\widetilde{S_{in}}$. For G in Fig. 1, such adjustment is not needed.

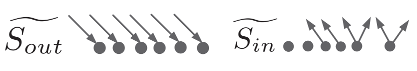


Figure 2: $\widetilde{S_{in}}, \widetilde{S_{out}}$ after node scaling for G in Fig.1.

Algorithm 2: SCALING(S_{in}, $\widetilde{n}$, $\widetilde{m}$)

```
1  node_scaling($S_{in}$, $\widetilde{n}$)
2  node_adjustment($\widetilde{S_{in}}$, $\widetilde{n}$, $S_{in}$)
   /* $h_0/l_0$ is the upper/lower bound where x
      varies. By default, $h_0/l_0$ should be the
      highest/lowest degree in the graph. $h_0/l_0$
      will be further extended if needed. t is the
      edge difference threshold where the loop
      stops. $ct(x)$ refers to $ct_{\widetilde{S_{in}}}(x)$.        */
3  initialize $h = h_0, l = l_0, t$
5  while $|\sum_x ct(x) \times x - \widetilde{m}| > t$ do
6      if $l>=h$ then
7          $l = l_0$; h = $h_0$;
8      if $\widetilde{m} > \sum_x ct(x) \times x$ then
9          if $ct(l)>0$ then
10             $ct(l) - -$; $ct(h) + +$;
11             $l + +$; $h - -$;
12         else $l + +$;
13     else
14         if $ct(h)>0$ then
15             $ct(l) + +$; $ct(h) - -$;
16             $l + +$; $h - -$;
17         else $h - -$;
18 adjust $|\sum_x ct(x) \times x - \widetilde{m}|$ edges
```

- **edge adjustment**. Next, the number of scaled edges must equal to $\widetilde{m}$, i.e. $\sum_x ct_{\widetilde{S_{in}}}(x) \times x = \widetilde{m}$.

  If $\sum_x ct_{\widetilde{S_{in}}}(x) \times x < \widetilde{m}$, we increase the number of high degree nodes, and decrease the number of low degree nodes. If $\sum_x ct_{\widetilde{S_{in}}}(x) \times x > \widetilde{m}$, we decrease the number of high degree nodes, and increase the number of low degree nodes. The details are shown in Algo. 2.

  In our running example, $\sum_x ct_{\widetilde{S_{in}}}(x) \times x = 6 < 7 = \widetilde{m}$. Hence, we increase the number of high degree nodes (indegree=2), and decrease the number of low degree (indegree=1) nodes. Thus, we have 1 node with indegree=2 and 5 nodes with indegree=1 for $\widetilde{S_{in}}$. After the edge adjustment, the correct $\widetilde{S_{in}}, \widetilde{S_{out}}$ are shown in Fig. 1.

3.3 NODE SYNTHESIS

Now we have $\widetilde{S_{in}}$ and $\widetilde{S_{out}}$, and we match 1 piece in $\widetilde{S_{in}}$ to 1 piece in $\widetilde{S_{out}}$ and merge them to give a larger piece. This synthesis follows a *bidegree* distribution $f_{bi} : N^2 \to [0, 1]$, where $f_{bi}(d_1, d_2) = z$ means a fraction z of nodes have bidegree (d_1, d_2). We say a node u has bidegree (d_1, d_2) if it has indegree=d_1 and outdegree=d_2. For G in Fig. 1, the corresponding f_{bi} is listed in Table 2.

GSCALER loops through $f_{bi}(d_1, d_2)$ to synthesize nodes. However, for a desired bidegree (d_1, d_2), $\widetilde{S_{in}}$ and $\widetilde{S_{out}}$ may not have the necessary pieces. Hence, a *neighboring* $(d_1{}', d_2{}')$ will be used. GSCALER uses a greedy heuristic that matches pieces by minimizing the Manhattan distance

$$||(d_1, d_2) - (d_1{}', d_2{}')||_1 = |d_1 - d_1{}'| + |d_2 - d_2{}'|.$$

Algorithm 3: NODE SYNTHESIS($\widetilde{S_{in}}, \widetilde{S_{out}}, f_{bi}$)

```
1 while $\widetilde{S_{in}}$ and $\widetilde{S_{out}}$ not empty do
2     for $f_{bi}(d_1, d_2)$ do
3         while $\lfloor f_{bi}(d_1, d_2) \times \widetilde{m} \rfloor > 0$ do
4             $(d_1', d_2') \leftarrow Manhattan(d_1, d_2)$
5             $\widetilde{S_{bi}} \leftarrow (d_1', d_2')$
6             update $\widetilde{S_{in}}$, $\widetilde{S_{out}}$, $f_{bi}(d_1, d_2)$
```

For Table 2, when GSCALER sees $f_{bi}(1,0) = \frac{1}{3}$, it will first generate 1 node with bidegree $(1,0)$, and generate the other node with bidegree $(1,1)$. Next, GSCALER sees $f_{bi}(1,1) = \frac{1}{3}$, two nodes both having bidegree $(1,1)$ are generated. Lastly, GSCALER sees $f_{bi}(1,2) = \frac{1}{3}$, and two nodes with bidegree $(1,2)$, and $(2,2)$ are generated.

Algo.3 summarizes the node synthesis. The synthesized pieces for $\widetilde{S_{bi}}$ in $\widetilde{G}$ are shown in Fig. 1. Note that each piece in $\widetilde{S_{bi}}$ maps to a node in $\widetilde{G}$.

3.4 EDGE SYNTHESIS

Now we are almost done with the graph scaling, we only need to link the edges. This is similar to a jigsaw puzzle, we only need to make sure that each piece links to another correctly. When linking the pieces from $\widetilde{S_{bi}}$, we link 1 outgoing edge from a source node $v_s \in \widetilde{S_{bi}}$ to 1 incoming edge from a target node $v_t \in \widetilde{S_{bi}}$. There are numerous ways of joining the nodes. GSCALER synthesizes edges based on the edge correlation function

$$f_{corr} : N^2 \times N^2 \to [0,1],$$

where $f_{corr}(\alpha_s, \alpha_t) = z$ means a fraction z of the edges have a source node with bidegree α_s and a target node with bidegree α_t. The f_{corr} for G is listed in Table 3.

Instead of synthesizing edges one by one based on f_{corr} directly, GSCALER undergoes *Correlation Function Scaling* to scale f_{corr} to $\widetilde{f_{corr}}$ for $\widetilde{G}$. After a suitable $\widetilde{f_{corr}}$ is found, GSCALER links edges based on $\widetilde{f_{corr}}$.

3.4.1 Correlation Function Scaling

GSCALER loops through f_{corr} to produce $\widetilde{f_{corr}}$. For each $f_{corr}(\alpha_s, \alpha_t)$, it does the following *Iterative Correlating*:

Manhattan Minimization
GSCALER chooses the ***closest*** (α_s', α_t') for $\widetilde{f_{corr}}$ by minimizing $||\alpha_s - \alpha_s'||_1 + ||\alpha_t - \alpha_t'||_1$. For $f_{corr}((1,2),(1,0)) = 1/3$ in Table 3, GSCALER chooses $(1,2)$ for α_s' and $(1,0)$ for α_t'.

Increment Probability Maximization
GSCALER increments $\widetilde{f_{corr}}(\alpha_s', \alpha_t')$ by p, where p needs to be the **largest** number that satisfies the following constraints:
C1. $p \le f_{corr}(\alpha_s, \alpha_t)$.
C2. $p \le \min\{\frac{O(\alpha_s')}{\widetilde{m}}, \frac{I(\alpha_t')}{\widetilde{m}}\}$, where $I(\alpha_t')$ is the total number of available *incoming* edges for nodes with bidegree α_t', and $O(\alpha_s')$ is the total number of available *outgoing* edges for nodes with bidegree α_s'. C2 guarantees incremented number of edges does not exceed the total available number of edges of source nodes and target nodes.
C3. $p \le \frac{ct_{\widetilde{S_{bi}}}(\alpha_s') \times ct_{\widetilde{S_{bi}}}(\alpha_t')}{\widetilde{m}} - \widetilde{f_{corr}}(\alpha_s', \alpha_t')$. C3 ensures the total number of edges from source nodes to target nodes is not more than the maximal number of edges allowed from source nodes to target nodes (*no multiple edges*).

$f_{bi}(1,0) = \frac{1}{3}$
$f_{bi}(1,1) = \frac{1}{3}$
$f_{bi}(1,2) = \frac{1}{3}$

Table 2: Bidegree distribution for G in Fig. 1.

$f_{corr}((1,2),(1,0)) = \frac{1}{3}$
$f_{corr}((1,2),(1,1)) = \frac{1}{3}$
$f_{corr}((1,1),(1,2)) = \frac{1}{3}$

Table 3: Edge Correlation for G in Fig.1.

For $f_{corr}((1,2),(1,0)) = 1/3$ in Table 3:
By **C1**, $p \le 1/3$.
By **C2**, $p \le \min\{\frac{O((1,2))}{7}, \frac{I((1,0))}{7}\} = \min\{\frac{2}{7}, \frac{1}{7}\} = \frac{1}{7}$.
By **C3**, $p \le \frac{ct_{\widetilde{S_{bi}}}((1,2)) \times ct_{\widetilde{S_{bi}}}((1,0))}{7} - \widetilde{f_{corr}}((1,2),(1,0)) = \frac{1\times 1}{7} - 0$.
Hence, the incremental value is $p = \min\{\frac{1}{3}, \frac{1}{7}, \frac{1}{7}\} = \frac{1}{7}$.

Value Update
Next, GSCALER updates the distributions:

- $\widetilde{f_{corr}}(\alpha_s', \alpha_t') \leftarrow \widetilde{f_{corr}}(\alpha_s', \alpha_t') + p$.
- $O(\alpha_s') \leftarrow O(\alpha_s') - p \times \widetilde{m}$.
- $I(\alpha_t') \leftarrow I(\alpha_t') - p \times \widetilde{m}$.

For $f_{corr}((1,2),(1,0)) = 1/3$ in Table 3, GSCALER gets $\widetilde{f_{corr}}((1,2),(1,0)) = 1/7$, $O((1,2)) = 1$, $I((1,0)) = 0$. Table 4 shows the resulting scaled $\widetilde{f_{corr}}$.

After iterative correlating, and due to the no multiple edges constraint, it is possible that $\sum \widetilde{f_{corr}}(\alpha_s', \alpha_t') < 1$, which we fix by **random swapping**. This swap first randomly permutes the leftover bidegree from I and O without violating C3, then takes one element with bidegree γ_s' from O and one element with bidegree γ_t' from I to swap with generated $\widetilde{f_{corr}}(\alpha_s', \alpha_t')$.

The idea is to break 1 edge from some source node v_s with bidegree α_s' to some target node v_t with bidegree α_t', and form 2 new edges: 1 edge pointing from some node with bidegree γ_s' to the other node with bidegree α_t', and 1 edge pointing from some node with bidegree α_s' to some node with bidegree γ_t'. If C3 allows, then update $\widetilde{f_{corr}}$ as follows:

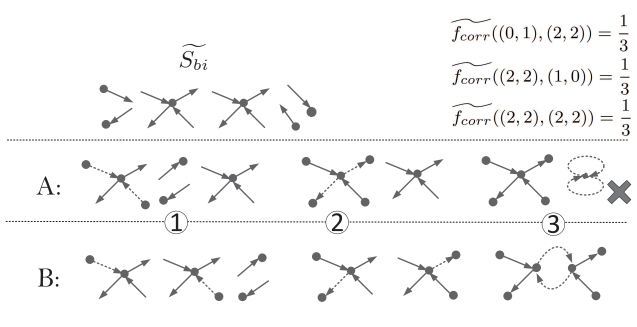
- $\widetilde{f_{corr}}(\alpha_s', \gamma_t') \leftarrow \widetilde{f_{corr}}(\alpha_s', \gamma_t') + \frac{1}{\widetilde{m}}$.
- $\widetilde{f_{corr}}(\gamma_s', \alpha_t') \leftarrow \widetilde{f_{corr}}(\gamma_s', \alpha_t') + \frac{1}{\widetilde{m}}$.
- $\widetilde{f_{corr}}(\alpha_s', \alpha_t') \leftarrow \widetilde{f_{corr}}(\alpha_s', \alpha_t') - \frac{1}{\widetilde{m}}$.

One successful swap thus increases $\widetilde{f_{corr}}$ by $\frac{1}{\widetilde{m}}$.

In the worst case (this did not happen in our experiments), after random swaps, $\sum \widetilde{f_{corr}}(\alpha_s', \alpha_t') < 1$ might still hold. We just leave $\widetilde{f_{corr}}$ as it is, and we will introduce some *dummy nodes* to link these $(1 - \sum \widetilde{f_{corr}}(\alpha_s', \alpha_t')) \times \widetilde{m}$ edges

$\widetilde{f_{corr}}((1,2),(1,0)) = \frac{1}{7}$	$\widetilde{f_{corr}}((1,2),(1,1)) = \frac{1}{7}$
$\widetilde{f_{corr}}((1,1),(1,2)) = \frac{1}{7}$	$\widetilde{f_{corr}}((2,2),(1,1)) = \frac{2}{7}$
$\widetilde{f_{corr}}((1,1),(2,2)) = \frac{2}{7}$	

Table 4: Edge Correlation for $\widetilde{G}$ in Fig. 1.



$\widetilde{f_{corr}}((0,1),(2,2)) = \frac{1}{3}$
$\widetilde{S_{bi}}$
$\widetilde{f_{corr}}((2,2),(1,0)) = \frac{1}{3}$
$\widetilde{f_{corr}}((2,2),(2,2)) = \frac{1}{3}$
A:
①
②
③
B:

Figure 3: Local linking of source and destination.

later in Sec.3.4.2.

3.4.2 Edge Linking

After $\widetilde{f_{corr}}$ is generated, GSCALER links the $p \times \widetilde{m}$ edges locally for each $\widetilde{f_{corr}}(\alpha_s{}', \alpha_t{}') = p$. Note that some ***linking*** steps are related, since $\alpha_s{}'$ might appear in a series of local linking steps, such as

$$\widetilde{f_{corr}}(\alpha_s{}', \alpha_{t_1}{}') = p_1,\ \ldots,\ \widetilde{f_{corr}}(\alpha_s{}', \alpha_{t_k}{}') = p_k$$

Hence, one must make sure that after $\widetilde{f_{corr}}(\alpha_s{}', \alpha_{t_1}{}') = p_1$ is done, $\widetilde{f_{corr}}(\alpha_s{}', \alpha_{t_2}{}') = p_2, \ldots, \widetilde{f_{corr}}(\alpha_s{}', \alpha_{t_k}{}') = p_k$ are still possible to be linked with no multiple edges. To emphasize the importance of this step, we use the following example to demonstrate both good and bad approaches.

In Fig.3, $\widetilde{S_{bi}}$ and $\widetilde{f_{corr}}$ are presented at the top. $\widetilde{S_{bi}}$ has 2 pieces of bidegree $(2,2)$, 2 pieces of bidegree $(1,0)$, and 2 pieces of bidegree $(0,1)$. The dotted edges represent newly linked edges at each step. If an edge has no source or target node attached, then it has not linked any two nodes yet. Example A demonstrates a failed strategy, while example B demonstrates a successful strategy.

For $\widetilde{f_{corr}}((0,1),(2,2)) = \frac{1}{3}$, it corresponds to local linking step ①, which links two edges from $(0,1)$ to $(2,2)$. Both examples link edges successfully. Example A links all 2 edges to 1 target node with bidegree $(2,2)$, whereas example B links 2 edges to 2 different target nodes with bidegree $(2,2)$.

For $\widetilde{f_{corr}}((2,2),(1,0)) = \frac{1}{3}$, it corresponds to local linking step ②, which links 2 edges from $(2,2)$ to $(1,0)$. Both examples link edges successfully. Example A links all 2 edges from 1 source node with bidegree $(2,2)$. Example B links 2 edges from 2 different source nodes with bidegree $(2,2)$.

For $\widetilde{f_{corr}}((2,2),(2,2)) = \frac{1}{3}$, it corresponds to local linking step ③, which links edges from $(2,2)$ to $(2,2)$. Example A produces multiple edges, which is not allowed, whereas example B successfully produces the graph.

Apart from the multiple edge constraint, the algorithm must be efficient as well. The naive idea of back tracking previous edge linking processes is obviously not practical for large graphs. GSCALER not only generates $\widetilde{G}$ with no multiple edges, it also links edges in **linear** time (see below). GSCALER first selects the proper *source* nodes L_s and *target* nodes L_t, and then link the edges from L_s to L_t.

Let $\mathcal{S}_{\alpha_s{}'}/\mathcal{T}_{\alpha_t{}'}$ be a *queue* of the source/target nodes with *bidegree* $\alpha_s{}'/\alpha_t{}'$ from $\widetilde{S_{bi}}$. For each $\widetilde{f_{corr}}(\alpha_s{}', \alpha_t{}') = p$, GSCALER dequeue&enqueue $p \times \widetilde{m}$ elements from $\mathcal{S}_{\alpha_s{}'}$ to L_s, and dequeue&enqueue $p \times \widetilde{m}$ elements from $\mathcal{T}_{\alpha_t{}'}$ to L_t. More specifically, whenever one element is dequeued from $\mathcal{S}_{\alpha_s{}'}/\mathcal{T}_{\alpha_t{}'}$, it will be put into L_s/L_t, and then been enqueued to $\mathcal{S}_{\alpha_s{}'}/\mathcal{T}_{\alpha_t{}'}$ again. For example, if we dequeue&enqueue 7 elements from $\mathcal{S}_{(1,1)} = [1,2,3,4,5]$ to L_s, then $L_s = [1,2,3,4,5,1,2]$, and $\mathcal{S}_{(1,1)} = [3,4,5,1,2]$. In general,

$$L_s = \{u_1^{\lceil \frac{p\times\widetilde{m}}{|\mathcal{S}_{\alpha_s{}'}|} \rceil}, \ldots, u_{r_s}^{\lceil \frac{p\times\widetilde{m}}{|\mathcal{S}_{\alpha_s{}'}|} \rceil}, u_{r_s+1}^{\lfloor \frac{p\times\widetilde{m}}{|\mathcal{S}_{\alpha_s{}'}|} \rfloor}, \ldots, u_{|\mathcal{S}_{\alpha_s{}'}|}^{\lfloor \frac{p\times\widetilde{m}}{|\mathcal{S}_{\alpha_s{}'}|} \rfloor}\}, \forall u_i \in \mathcal{S}_{\alpha_s{}'}$$

$$L_t = \{v_1^{\lceil \frac{p\times\widetilde{m}}{|\mathcal{T}_{\alpha_t{}'}|} \rceil}, \ldots, v_{r_t}^{\lceil \frac{p\times\widetilde{m}}{|\mathcal{T}_{\alpha_t{}'}|} \rceil}, v_{r_t+1}^{\lfloor \frac{p\times\widetilde{m}}{|\mathcal{T}_{\alpha_t{}'}|} \rfloor}, \ldots, v_{|\mathcal{T}_{\alpha_t{}'}|}^{\lfloor \frac{p\times\widetilde{m}}{|\mathcal{T}_{\alpha_t{}'}|} \rfloor}\}, \forall v_j \in \mathcal{T}_{\alpha_t{}'}$$

$$r_s = p \times \widetilde{m} \mod |\mathcal{S}_{\alpha_s{}'}|\ ,\ r_t = p \times \widetilde{m} \mod |\mathcal{T}_{\alpha_t{}'}|, \tag{4}$$

where v_1^k means v_1 appears k times in L_t, and k is called the ***multiplicity*** of v_1.

After L_s and L_t are generated, GSCALER links edges from L_s to L_t. Before that, we prove a theorem for ***Compound Multiplicity Reduction***; we later use this to show GSCALER does not generate multiple edges between two nodes.

THEOREM 1. *[Compound Multiplicity Reduction]. Given multisets* $\mathbb{U} = \{u_1^{m_{u_1}}, \ldots, u_{\theta_0}^{m_{u_{\theta_0}}}\}$, $\mathbb{V} = \{v_1^{m_{v_1}}, \ldots, v_{\theta_1}^{m_{v_{\theta_1}}}\}$,

$$\sum_{u_i \in \mathbb{U}} m_{u_i} = |\mathbb{U}| = |\mathbb{V}| = \sum_{v_j \in \mathbb{V}} m_{v_j}, \tag{5}$$

$$\max_{u_i \in \mathbb{U}} m_{u_i} - \min_{u_i \in \mathbb{U}} m_{u_i} \le 1, \tag{6}$$

$$\text{and } \max_{v_j \in \mathbb{V}} m_{v_j} - \min_{v_j \in \mathbb{V}} m_{v_j} \le 1, \tag{7}$$

there exists a multiset $\mathbb{W} = \{w_k | w_k(0) \in \mathbb{U}, w_k(1) \in \mathbb{V}\}$

$$\text{such that } \mathbb{U} = \bigcup_k \{w_k(0)\},\ \mathbb{V} = \bigcup_k \{w_k(1)\}, \tag{8}$$

$$\max_{w_k \in \mathbb{W}} m_{w_k} - \min_{w_k \in \mathbb{W}} m_{w_k} \le 1, \tag{9}$$

$$\forall w_k \in \mathbb{W}, m_{w_k} \le \lceil \frac{|\mathbb{W}|}{\theta_0 \times \theta_1} \rceil. \tag{10}$$

PROOF. Reorder $\mathbb{U}, \mathbb{V}$ to
$\mathbb{U} = (u_1, \ldots, u_1, u_2, \ldots, u_2, \ldots, u_{\theta_0}, \ldots, u_{\theta_0})$,
$\mathbb{V} = (v_1, v_2, \ldots, v_{\theta_1}, v_1, v_2, \ldots, v_{\theta_1}, v_1, \ldots)$.
Construct $\mathbb{W}$ as follows: $\forall w_k \in \mathbb{W}$, $w_k = (\mathbb{U}(k), \mathbb{V}(k))$, where $\mathbb{U}(k)$ means the kth element in $\mathbb{U}$. Therefore, Eq.8 is satisfied. Further, for any u_i, let
$\mathbb{W}_{u_i} = ((u_i, \mathbb{V}(c_i)), (u_i, \mathbb{V}(c_i+1)), \ldots, (u_i, \mathbb{V}(d_i)))$
be the sequence of all elements in $\mathbb{W}$ containing u_i as first coordinate. Consider $\mathbb{V}_{u_i} = (\mathbb{V}(c_i), \mathbb{V}(c_i+1), \ldots, \mathbb{V}(d_i))$, a sequence of elements in $\mathbb{V}$ pairing with u_i. Note $\mathbb{V}_{u_i}$ is a *periodic sequence* with *period*=θ_1. Thus, the maximum multiplicity in $\mathbb{V}_{u_i}$ is $\lceil \frac{|\mathbb{V}_{u_i}|}{\theta_1} \rceil$, and similarly for $\mathbb{W}_{u_i}$. Hence, the maximum multiplicity of $\mathbb{W}$ is

$$\max_{w_k \in \mathbb{W}} m_{w_k} = \max_{1 \le i \le \theta_0} \lceil \frac{|\mathbb{V}_{u_i}|}{\theta_1} \rceil \tag{11}$$

It is trivial that

$$|\mathbb{V}_{u_i}| = m_{u_i}, \forall i, 1 \le i \le \theta_0 \tag{12}$$

Moreover, by Eq.6, we will have

$$\forall u_i \in \mathbb{U}, m_{u_i} \le \lceil \frac{|\mathbb{U}|}{\theta_0} \rceil \tag{13}$$

Hence, by Eq.11, Eq.12, Eq.13, we will have

$$\max_{w_k \in \mathbb{W}} m_{w_k} = \max_{1 \le i \le \theta_0} \lceil \frac{|\mathbb{V}_{u_i}|}{\theta_1} \rceil \le \lceil \frac{\lceil \frac{|\mathbb{U}|}{\theta_0} \rceil}{\theta_1} \rceil = \lceil \frac{|\mathbb{U}|}{\theta_0 \times \theta_1} \rceil \tag{14}$$

Since $|\mathbb{U}| = |\mathbb{W}|$, therefore

$$\max_{w_k \in \mathbb{W}} m_{w_k} \le \lceil \frac{|\mathbb{W}|}{\theta_0 \times \theta_1} \rceil, \quad (15)$$

so Eq.10 holds. Similarly, the minimum multiplicity of $\mathbb{W}$ is

$$\min_{w_k \in \mathbb{W}} m_{w_k} = \min_{1 \le i \le \theta_0} \lfloor \frac{|\mathbb{V}_{u_i}|}{\theta_1} \rfloor \ge \lfloor \frac{|\mathbb{W}|}{\theta_0 \times \theta_1} \rfloor \quad (16)$$

Eq.9 follows from Eq.15 and Eq.16. We call such a $\mathbb{W}$ a **compound multiset**. □

For edge linking between L_s and L_t, GSCALER links $u_1 \in L_s$ to $v_2 \in L_t$ to form one edge (u_1, v_2) in L_e (edge set). By Eq.4, L_s/L_t satisfies $\mathbb{U}/\mathbb{V}$ in *Theorem 1* respectively, and it is easy to see that L_e is the compound $\mathbb{W}$ in *Theorem 1.*

THEOREM 2. *Given non-empty $L_s(\mathbb{U})$ dequeued&enqueued from $\mathcal{S}_{\alpha_s{}'}$, and non-empty $L_t(\mathbb{V})$ dequeued&enqueued from $\mathcal{T}_{\alpha_t{}'}$, the maximum multiplicity for edge set $L_e(\mathbb{W})$ as described in Theorem 1 is* 1.

PROOF. If $|L_s| \le |\mathcal{S}_{\alpha_s{}'}|$, then every element in L_s is unique. Hence, the maximum multiplicity for elements in L_s is 1, so L_e has maximum multiplicity of 1. The case is similar for $|L_t| \le |\mathcal{T}_{\alpha_t{}'}|$.

If $|L_s| > |\mathcal{S}_{\alpha_s{}'}|$ and $|L_t| > |\mathcal{T}_{\alpha_t{}'}|$, then L_s has $|\mathcal{T}_{\alpha_s{}'}|$ distinct elements, and L_t has $|\mathcal{T}_{\alpha_t{}'}|$ distinct elements. By *Theorem 1*, the maximum multiplicity of elements in L_e is

$$\lceil \frac{|L_e|}{|\mathcal{S}_{\alpha_s{}'}| \times |\mathcal{T}_{\alpha_t{}'}|} \rceil \quad (17)$$

Since $|L_s| = |L_e|$, and L_s has $\widetilde{f_{corr}}(\alpha_s{}', \alpha_t{}') \times \widetilde{m}$ elements, the maximum multiplicity of elements in L_e is

$$\lceil \frac{\widetilde{f_{corr}}(\alpha_s{}', \alpha_t{}') \times \widetilde{m}}{|\mathcal{S}_{\alpha_s{}'}| \times |\mathcal{T}_{\alpha_t{}'}|} \rceil$$

Moreover, by C3 in Sec.3.4.1, we know that

$$\widetilde{f_{corr}}(\alpha_s{}', \alpha_t{}') \times \widetilde{m} \le |\mathcal{S}_{\alpha_s{}'}| \times |\mathcal{T}_{\alpha_t{}'}|, \quad (18)$$

$$\text{so } \lceil \frac{|L_e|}{|\mathcal{S}_{\alpha_s{}'}| \times |\mathcal{T}_{\alpha_t{}'}|} \rceil = \lceil \frac{\widetilde{f_{corr}}(\alpha_s{}', \alpha_t{}') \times \widetilde{m}}{|\mathcal{S}_{\alpha_s{}'}| \times |\mathcal{T}_{\alpha_t{}'}|} \rceil \le 1 \quad (19)$$

□

Hence, by *Theorem 2*, there are no multiple edges from L_s to L_t. Therefore, GSCALER successfully produces a graph without multiple edges.

Moreover, the generation runs in **linear** time: For each $\widetilde{f_{corr}}(\alpha_s{}', \alpha_t{}')$, the generation for L_s and L_t is in linear time: After L_s, L_t are generated, the L_e is formed by sequentially matching the elements in L_s and L_t as described in *Theorem 1.* Hence, the total edge linking time is *linear.* Given the edge correlation $\widetilde{f_{corr}}$ in Table 4, GSCALER generates $\widetilde{G}$ as presented in Fig. 1.

Finally, as stated at the end of Sec.3.4.1, there might be some small possibility that $\sum \widetilde{f_{corr}}(\alpha_s{}', \alpha_t{}') < 1$ holds. This is the theoretical worst case (which has not happened in our experiments). To resolve this, GSCALER introduces ϵ *dummy nodes* into $\widetilde{G}$ for the purpose of maintaining degree distribution similarity. After the dummy nodes are introduced, all the $1 - \sum \widetilde{f_{corr}}(\alpha_s{}', \alpha_t{}')$ edges are linked to

Algorithm 4: EDGE SYNTHESIS($\widetilde{S_{bi}}$, f_{corr})

```
1 $\widetilde{f_{corr}} \leftarrow$ correlation_function_scaling($\widetilde{S_{bi}}$, $f_{corr}$)
2 while $\widetilde{f_{corr}}(\alpha_s{}', \alpha_t{}') = p$ do
      /* produce the $L_s$ and $L_t$ */
3     $L_s, L_t \leftarrow local\_set(\alpha_s{}', \alpha_t{}', p, \widetilde{m})$
      /* link the edges from $L_s$ to $L_t$ */
4     $\widetilde{G} \leftarrow edge\_linking(L_s, L_t)$
5 add dummy nodes to $\widetilde{G}$ (if necessary)
```

these dummy nodes. To avoid multiple edges, we can set $\epsilon = \max_\alpha\{I(\alpha), O(\alpha)\}$; this ϵ is obviously not the smallest possible. However, such an ϵ is already small(negligible) compared to $|\widetilde{G}|$, and such a scenario is rare. Algo.4 summarizes the edge synthesis.

4. EVALUATION

We first review the graph properties that we use for similarity measurement.

4.1 Graph Properties

There are numerous graph properties that can be chosen as the similarity measurement criteria, e.g. degree distribution, diameter, k-core distribution, etc. We choose the 7 most common graph properties used in the literature. Let $N = \{0, 1, 2, 3, \ldots\}$ and consider the following local and global graph properties:

1. **Indegree distribution** $f_{in} : N \to [0, 1]$
   $f_{in}(d) = z$ means a portion z of nodes have indegree d.

2. **Outdegree distribution** $f_{out} : N \to [0, 1]$
   $f_{out}(d) = z$ means a portion z of nodes have outdegree d.

3. **Bidegree distribution** $f_{bi} : N^2 \to [0, 1]$
   Defined in Sec 3.3.

4. **Ratio of largest strongly connected component (SCC)**
   An SCC of G is a maximal set of nodes such that for every node pair u and v, there is a directed path from u to v and another from v to u. The ratio is the number of nodes in the SCC divided by $|V|$.

5. **Average clustering coefficient (CC)**
   For node v_i, let N_i be the set of its neighbors. The local clustering coefficient [34] C_i for nodes v_i is defined by

$$C_i = \frac{|\{(v_j, v_k) : v_j, v_k \in N_i, (v_j, v_k) \in E\}|}{|N_i|(|N_i| - 1)}$$

   The average clustering coefficient [15] $\bar{C} = \frac{\sum_{v_i \in V} C_i}{|V|}$

6. **Average shortest path length (ASPL)**
   For u and v in V, the *pairwise distance* $d(u, v)$ is the number of edges in the shortest path from u to v; $d(u, v) = \infty$ iff there is no path from u to v (where ∞ is some number greater than $|E|$). The ASPL is

$$\frac{\sum_{d(u,v)<\infty} d(u, v)}{|V| \times (|V| - 1)}.$$

7. **Effective diameter** [21]
The effective diameter is the smallest $k \in N$ that is greater than 90% of all pairwise distances that satisfy $d(u,v) < \infty$.

4.2 Measuring Similarity

For a scalar graph property α, let α_G denote the α value for G and $\alpha^{(i)}_{\widetilde{G}}$ denote the α value for $\widetilde{G}$ constructed with algorithm $\mathcal{A}^{(i)}$. We can compare $\mathcal{A}^{(1)}$ and $\mathcal{A}^{(2)}$ by the absolute difference $|\alpha^{(i)}_{\widetilde{G}} - \alpha_G|$ or the relative difference $|\alpha^{(i)}_{\widetilde{G}} - \alpha_G|/\alpha_G$ These are equivalent since

$$|\alpha^{(1)}_{\widetilde{G}} - \alpha_G| < |\alpha^{(2)}_{\widetilde{G}} - \alpha_G| \Longleftrightarrow \frac{|\alpha^{(1)}_{\widetilde{G}} - \alpha_G|}{\alpha_G} < \frac{|\alpha^{(2)}_{\widetilde{G}} - \alpha_G|}{\alpha_G}$$

However, an α like effective diameter is an integer, whereas a property like average clustering coefficient has $\alpha < 1$. Some information on α_G is thus lost if we plot relative differences, so we will plot absolute differences instead.

For a degree distribution $f : N^d \to [0,1]$, we follow Leskovec and Faloutsos [21] and use a Kolmogorov-Smirnov (KS) D-statistic to measure the difference between distributions f_G and $f_{\widetilde{G}}$. For $d = 1$, the statistic is defined as

$$\sup_x |F_G(x) - F_{\widetilde{G}}(x)|$$

where F_G and $F_{\widetilde{G}}$ are the cumulative distribution functions (cdf) for f_G and $f_{\widetilde{G}}$. For $d > 1$, defining the cdf is not straightforward, since there are $2^d - 1$ possibilities. In this paper, we adopt Fasano and Franceschini's computationally efficient variant [10] of the KS D-statistic.

4.3 Algorithms

We compare GSCALER to state-of-art algorithms (for graph sampling, generative models, and database scaling) that have been widely used as baselines for comparison.

- In Random Walk with Escaping (RW), a starting node v_0 is chosen uniformly at random. Each step in the random walk samples an unvisited neighbor uniformly at random, and there is a probability 0.8 (following [21]) that the walk restarts at v_0. If the walk reaches a dead end, the walk restarts with a new v_0.

- In Forest Fire (FF) [21], a v_0 is similarly chosen. At each step, first choose a positive integer y that is geometrically distributed with mean $p_f(1-p_f)$, and let $x = \lceil y \times c \rceil$, $c \in [0,1]$. (We follow Leskovec and Faloutsos and set $p_f = 0.7$ and $c = 1$.) Pick uniformly at random neighbors $w_1, \ldots, w_x$ that are not yet sampled, then recursively repeat these steps at each w_i.

- In Stochastic Kronecker Graph (SKG) [20], SKG first trains a N_1 by N_1 matrix K_1, where N_1 is typically set as 2. Then recursively multiply K_1 through *Kronecker* product. Thus, d multiplication of Kronecker product results in a graph of N_1^d nodes. In our experiment, we use the N_1^d closest to $\widetilde{n}$ as the target number $\widetilde{n}$.

- *UpSizeR* was designed to synthetically scale a relational dataset by maintaining the degree distribution [33]. However, *UpSizeR* does not allow free choice of $\widetilde{m}$. For our experiments, we transform the graph to relational tables so *UpSizeR* can scale it.

4.4 Datasets

We pick 2 real directed graphs from Stanford's collection of networks [24]. They are large enough that it makes sense for scaling down, but small enough for global properties like diameter to be determined in reasonable time. These two graphs were also used by previous authors [20–22, 25].

- *Epinions* ($|V| = 75879$, $|E| = 508837$) for the website `Epinions.com`, where an edge (x, y) indicates user x trusts user y.

- *Slashdot* is a website for technology-related news, where users tag others as friend or foe. The graph contains friend/foe links between the users of Slashdot. The dataset *Slashdot0811* ($|V| = 77360$, $|E| = 828161$) was from November 2008.

Given the space constraint, we choose to compare more properties for 2 graphs, instead of comparing fewer properties for more graphs.

5. RESULTS AND DISCUSSION

All experiments are done on a Linux machine with 128GB memory and AMD Opteron 2.3GHz processor. For *SKG*, we use the C++ implementation [24]. For *UpSizeR*, we use the authors' C++ implementation [1]. For *FF, RW* and GSCALER, we implemented them in Java.

These algorithms use the number of nodes n to specify sample size or scale factor, so we define $s = \widetilde{n}/n$. In our experiments, we set $s = \frac{1}{5}, \frac{1}{7.5}, \frac{1}{10}, \frac{1}{12.5}, \frac{1}{15}, \frac{1}{17.5}, \frac{1}{20}$ for scaling down, and $s = 2, 3, 4, 5, 6$ for scaling up.

GSCALER allows the user to specify the number of edges $\widetilde{m}$. However, an $\widetilde{m}$ that is arbitrarily small or arbitrarily large will make it impossible for $\widetilde{G}$ to be similar to G. To be fair, we choose an $\widetilde{m}$ for GSCALER that yields the best results. We will revisit this issue in Sec.6.

For each s value, we run each algorithm 10 times (using different seeds) on each dataset. The average of these 10 runs is then plotted as one data point.

5.1 Execution Time

For a fair comparison, we exclude the I/O time for all algorithms.

Execution time for *SKG* has two parts: training time and running time. *SKG* needs to train the graph initiator matrix K_1, where K_1 is a 2 by 2 matrix in our case; K_1 can be pre-computed. After K_1 is trained, *SKG* uses K_1 to generate a graph with 2^k nodes.

To get a better understanding of *SKG*'s time complexity, we plot both training time and running time. *SKG-Train* represents the time needed for training K_1, while *SKG-Run* represents the running time of graph generation given K_1.

Fig.4 and Fig.5 show the execution time for all algorithms using log scale.

For both datasets, SKG-Train is very large for training the graph initiator K_1. However, after K_1 is trained, graph generation is fast (seconds), so SKG-Run is small. That aside, GSCALER has the smallest execution time, which is faster than SKG-Train by about 2 orders of magnitude, and faster than *RW*, *FF*, *UpSizeR* by about 1 order of magnitude.

[1] `http://www.comp.nus.edu.sg/~upsizer/`

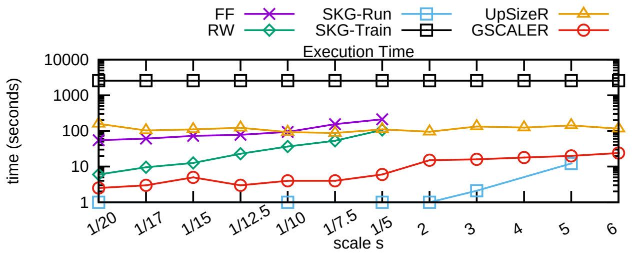


Figure 4: Execution time (log scale) for *Slashdot*. FF and RW do not work for $s > 1$.

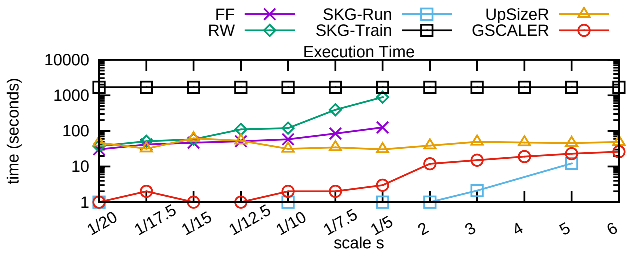


Figure 5: Execution time (log scale) for *Epinions*. FF and RW do not work for $s > 1$.

5.2 Theoretical Bounds of Degree Distribution

Before looking at the similarity comparison, we first show theoretically that GSCALER performs well on varying $\widetilde{m}$ and $\widetilde{n}$ for indegree/outdegree distribution. To save space, we just consider indegree distribution throughout this section. The proof for the outdegree distribution is similar.

THEOREM 3. *Given original graph G's indegree distribution is f_{in}, and GSCALER scales G to $\widetilde{G}$, where $\widetilde{m}$ and $\widetilde{n}$ scale by the same ratio s. Then,* $\text{E}[D(f_{in}, \widetilde{f_{in}})] = 0$.

PROOF. As explained in Sec.3.2, GSCALER scales S_{in} with the following criterion:

$$\text{E}[ct_{\widetilde{S_{in}}}(x)] = ct_{S_{in}}(x) \times \tfrac{\widetilde{n}}{n}$$

Now, consider the current scaled number of nodes $|\widetilde{V}|$ and number of edges $|\widetilde{E}|$. After node scaling,

$$\begin{aligned}\text{E}[|\widetilde{V}|] &= \text{E}[\sum_x ct_{\widetilde{S_{in}}}(x)] = \sum_x \text{E}[ct_{\widetilde{S_{in}}}(x)] = \sum_x ct_{S_{in}}(x) \times s \\ &= n \times s = \widetilde{n}.\\ \text{E}[|\widetilde{E}|] &= \text{E}[\sum_x ct_{\widetilde{S_{in}}}(x) \times x] = \sum_x \text{E}[ct_{\widetilde{S_{in}}}(x) \times x] \\ &= \sum_x \text{E}[ct_{\widetilde{S_{in}}}(x)] \times x = \sum_x ct_{S_{in}}(x) \times s \times x \\ &= \sum_x ct_{S_{in}}(x) \times x \times s = m \times s = \widetilde{m}.\end{aligned}$$

Hence, no *edge adjustment* is expected in Sec.3.2. Thus,

$$\forall x, \text{E}[\widetilde{f_{in}}(x) - f_{in}(x)] = 0.$$

Consequently, $\text{E}[D(f_{in}, \widetilde{f_{in}})] = 0$. $\square$

However, a GSCALER user may want to have a $\widetilde{G}$ with average degree different from G, i.e. $\widetilde{m}$ and $\widetilde{n}$ scale by different factors. In this case, GSCALER can still produce a similar degree distribution with small and bounded error.

THEOREM 4. *Given an original graph G's indegree distribution is f_{in}, and GSCALER scales G to $\widetilde{G}$, where $\widetilde{n} = n \times s$, $\widetilde{m} = m \times s \times (1 + r)$ and $r \neq 0$. Then,*

$$\text{E}[D(f_{in}, \widetilde{f_{in}})] \leq \frac{2m|r|}{n \times d^*}$$

where d^* is approximately the largest degree of G.

PROOF. As shown in the proof of *Theorem 3*, after node scaling, $\text{E}[|\widetilde{V}|] = n \times s = \widetilde{n}$ and $\text{E}[|\widetilde{E}|] = m \times s \neq \widetilde{m}$.

Hence, *edge adjustment* is needed, as stated in Sec.3.2. In total, we are expecting $m \times s \times (1 + r)$ edges. Hence, $|m \times rs|$ edges are expected to be added/removed. We refer to $ct_{\widetilde{S_{in}}}(x)$ as $ct(x)$ if there is no ambiguity.

Consider $r > 0$, so $m \times rs$ edges are to be added. This is done by the edge adjustment operation as stated in Algo.2: (i) $ct(l) - -$, $ct(h) + +$, (ii) $l + +$, $h - -$. The net effect is to add $h - l$ edges per adjustment.

Let d_m be the maximum degree of G. We assume l is initiated as 0, while h is initialized as d_m.
Case $ct(l) > 0$ for all the first k adjustment:
Then GSCALER will add

$$d_m, d_m - 2, d_m - 4 \ldots, d_m - [(k-1) \mod \lceil \frac{d_m}{2} \rceil] \times 2 \quad (20)$$

edges for the first k adjustments. Let T_k be the total number of edges added by first k adjustments, then

$$T_k \geq \frac{d_m}{2} \times k \quad (21)$$

Since $m \times rs$ edges are expected to be added, the expected number of adjustments k satisfies

$$k \leq \frac{m \times rs}{\frac{d_m}{2}} = \frac{2m \times rs}{d_m} \quad (22)$$

Moreover, each edge adjustment changes $\widetilde{f_{in}}$ by
(i) decrementing $\frac{1}{\widetilde{n}}$ for some $\widetilde{f_{in}}(x_i)$, where $x_i < \frac{d_m}{2}$
(ii) incrementing $\frac{1}{\widetilde{n}}$ for some $\widetilde{f_{in}}(x_j)$, where $x_j > \frac{d_m}{2}$
Thus, the expected total decremental changes made to $\widetilde{f_{in}}$ is $\frac{1}{\widetilde{n}} \times k$. By Eq.22,

$$\frac{1}{\widetilde{n}} \times k \leq \frac{1}{\widetilde{n}} \times \frac{2m \times rs}{d_m} = \frac{2m \times rs}{sn \times d_m} = \frac{2mr}{n \times d_m} \quad (23)$$

Since $D(f_{corr}, \widetilde{f_{corr}})$ measures largest difference between the cumulative function of $f_{corr}, \widetilde{f_{corr}}$. Hence, by Eq.23,

$$\text{E}[D(f_{corr}, \widetilde{f_{corr}})] \leq \frac{2mr}{n \times d^*}, \text{ where } d^* = d_m \quad (24)$$

Case $\exists l_j, ct(l_j) = 0$. Assume at the ith adjustment, $ct(l_0) = 0$ for some l_0, where i is the smallest.
Then GSCALER shifts l to l_0+1, and try to decrease $ct(l_0+1)$. Assume $ct(l_0 + 1) > 0$, then we can do $ct(l_0 + 1) - -$.

1. $l_0 + 1 \neq h$: By Eq.20, the number of edges added at ith step is $d_m - [(i-1) \mod \lceil \frac{d_m}{2} \rceil] \times 2 - 1$. Hence, $T_i = T_{i-1} + d_m - [(i-1) \mod \lceil \frac{d_m}{2} \rceil] \times 2 - 1$. By Eq.21,

$$T_i + 1 \geq \frac{d_m}{2} \times i \quad (25)$$

Then, $T_i \geq \frac{d_m}{2} \times i - 1 = \frac{d_m - 1}{2} \times i + (\frac{i}{2} - 1)$
Since $T_1 = d_m - 1 \geq \frac{d_m - 1}{2}$, then $T_i \geq \frac{d_m - 1}{2} \times i \; \forall i \in N$

2. $l_0 + 1 = h$: Then l will be reset to 0, and h will be reset to d_m, so the number of edges at ith adjustment is obviously larger than $d_m - [(i-1) \mod \lceil \frac{d_m}{2} \rceil] \times 2 - 1$. Hence, Eq.25 will hold as well.

Therefore, during the first k adjustments, if we encounter w different $l_0, l_1, \ldots, l_{w-1}$, such that $ct(l_j) = 0, \forall 0 \leq j < w$, then by mathematical induction, one can still conclude that

$$T_i \geq \frac{d_m - w}{2} \times i.$$

Hence, let $d^* = d_m - w$ and follow the proof after Eq.21; then $\mathrm{E}[D(f_{corr}, \widetilde{f_{corr}})] \leq \frac{2mr}{n \times d^*}$, where d^* is close to d_m in general.

The proof for $r < 0$ is similar $\square$

5.3 Experimental Results

All figures in this section (except Figs.7, 9, 11,12) plot the similarity measures (Sec. 4.2) comparing G and $\widetilde{G}$, where

- the horizontal axis is scale factor $s = \frac{\tilde{n}}{n}$;
- the vertical axis is KS D-statistic for indegree, outdegree and bidegree distributions;
- the vertical axis is absolute error for the other 5 properties (effective diameter, largest SCC ratio, etc.);
- the true value for all datasets' graph properties are provided above each figure;
- we choose $\widetilde{m}$ to give best results for GSCALER; the other algorithms do not allow a free choice of $\widetilde{m}$.

We first look at *scaling down* for all the algorithms. As mentioned in Sec.1, scaling down in GSP has an objective that is different from graph sampling. However, we use graph sampling algorithms for comparison because GSP is a new problem, so there is no other algorithms for comparison except *UpSizeR* and *SKG*.

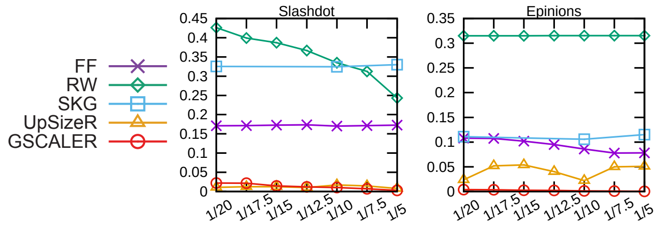


Figure 6: KS-D statistics for Indegree Distribution

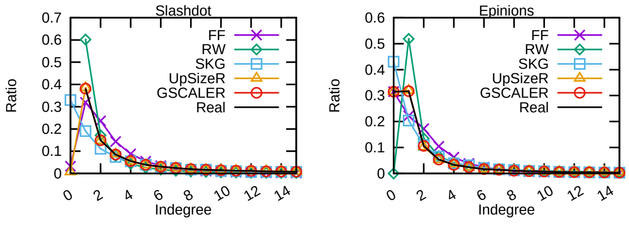


Figure 7: Indegree Distribution Plot

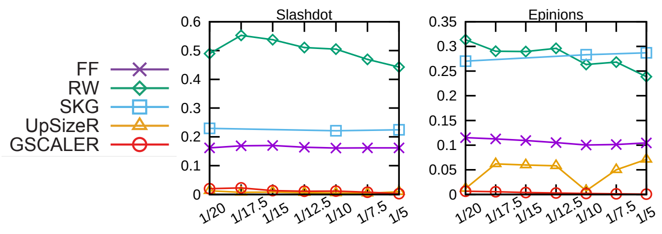


Figure 8: KS-D statistics for Outdegree Distribution

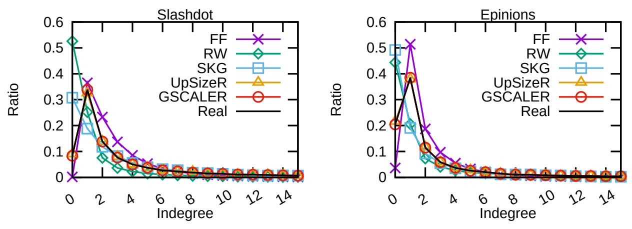


Figure 9: Outdegree Distribution Plot

5.3.1 *Indegree Distribution*

For the indegree distribution, Fig. 6 shows that both GSCALER and *UpSizeR* perform very well for *Slashdot*, with error < 0.01. For *Epinions*, *UpSizeR* has an error of 0.05 on average, whereas GSCALER again has a small error < 0.01.

For more details, Fig. 7 plots the indegree distribution for $s = 0.2$, where the x-axis is the indegree, and the y axis is the ratio of nodes having that indegree. We only show the plot up to *indegree* $= 15$, which covers more than 87% of all nodes.

The plot shows that GSCALER and *UpSizeR* mimics G's indegree distribution very well. Although all algorithms produce power law shaped distributions, only *UpSizeR* and GSCALER give a close fit for the empirical distribution.

5.3.2 *Outdegree Distribution*

For the outdegree distribution, Fig. 8 similarly shows that both GSCALER and *UpSizeR* perform very well for *Slashdot*, with error < 0.03 on average. For *Epinions*, *UpSizeR* has an error of 0.05 on average, whereas GSCALER still has an error < 0.01.

The outdegree distributions are plotted in Fig. 9. We observe that GSCALER and *UpSizeR* also closely match G's outdegree distribution. Again, although all algorithms produce power law shaped distributions, *UpSizeR* and GSCALER give the best fit for the empirical outdegree distribution.

5.3.3 *Bidegree Distribution*

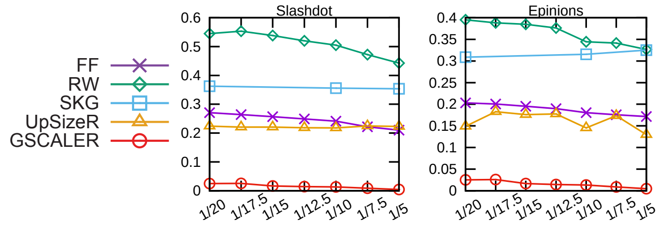


Figure 10: KS-D statistics for Bidegree Distribution

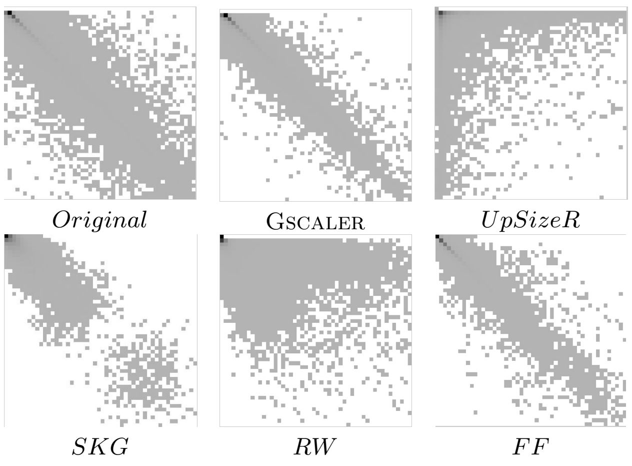


Figure 11: Bidegree distribution for *Slashdot*

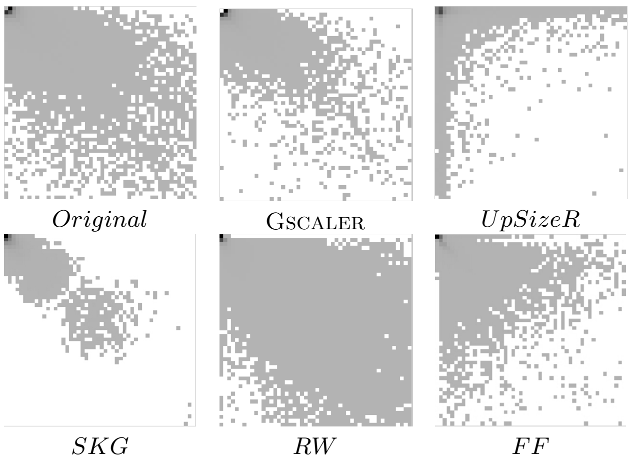


Figure 12: Bidegree distribution for *Epinions*

Even though *UpSizeR* matches the empirical indegree and outdegree distributions, it is not able to capture the bidegree distribution that describes the correlation between indegree and outdegree. Such correlation is especially important for social network graphs. For example, in *Epinions*, the number of people he/she trusts and the number of people who trust him/her are correlated. As shown in Fig.10, only GSCALER captures such correlation very well.

For a detailed look at the bidegree distributions, we give 2-dimensional plots Fig. 11 and Fig. 12. We transform the bidegree distribution to a $k \times k$ matrix B. Each cell i, j represents the portion of nodes with bidegree (i, j). In other words, $B[i, j] = f_{bi}(i, j)$. When visualizing B, we use a gray scale intensity plot for cell i, j to indicate $B[i, j]$. The larger f_{bi} is, the darker the cell (i, j) is. In our case, we set $k = 50$ which covers more than 90% of total nodes.

Fig. 11 and Fig. 12 show *indegree* is positively correlated to *outdegree*. For both *Slashdot* and *Epinions*, GSCALER is the best algorithm in capturing this indegree/outdegree correlation. We also observe that *UpSizeR* tends to have indegree and outdegree negatively correlated. *SKG* has very concentrated and similar-shaped plots for both datasets. We suspect this is because of the self-similar matrix operation, Kronecker product. For *FF*, it captures the bidegree correlation for *Slashdot*'s bidegree, but not for *Epinions*.

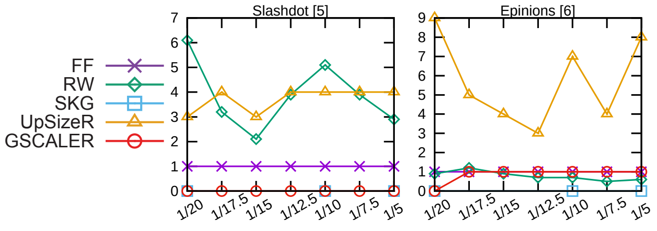


Figure 13: Absolute Error for Effective Diameter

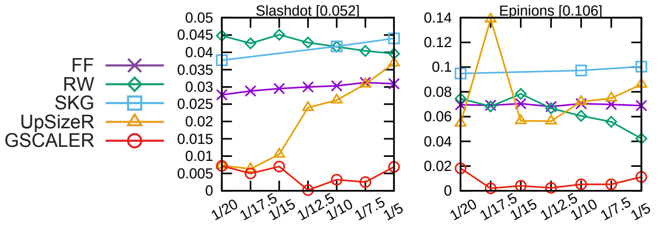


Figure 14: Absolute Error for Average CC

5.3.4 *Effective Diameter*

Fig. 13 shows that, for *Slashdot*, GSCALER produces exactly the real effective diameter; for *Epinions*, it produces an effective diameter with an absolute error no larger than 1. Overall, GSCALER, *FF*, *SKG* are the best algorithms in producing similar effective diameters.

5.3.5 *Average Clustering Coefficient*

For both datasets, GSCALER significantly reduces the error for average clustering coefficient. Fig. 14 shows that, for *Slashdot*, the average error for the other algorithms are between 0.03 and 0.045. This corresponds to a relative error of 60% to 90%. However, GSCALER only has an absolute error 0.005 on average, which corresponds to a relative error $< 10\%$. Similarly, for *Epinions*, GSCALER also improves the relative error from 70% to 10% on average.

5.3.6 *Largest SCC Ratio*

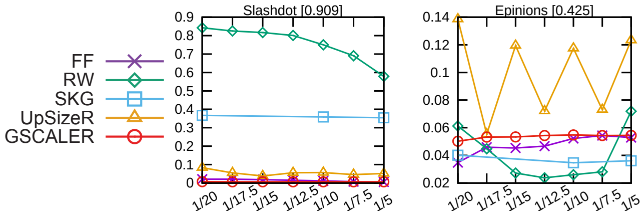


Figure 15: Absolute Error for Largest SCC Ratio

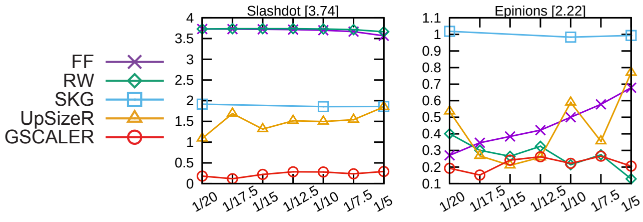


Figure 16: Absolute Error for ASPL

For largest SCC ratio, Fig. 15 shows that FF, *UpSizeR*, and GSCALER have the best performance for *Slashdot*. For *Epinions*, SKG is the best performing algorithm, whereas GSCALER only loses to SKG by an error < 0.01.

Note that, GSCALER and FF are the best performing algorithms which produce best similar largest SCC ratio for both datasets on average.

5.3.7 Average Shortest Path Length

Fig. 16 shows that GSCALER is the best algorithm in producing similar $ASPL$ for the scaled graph.

For *Slashdot*, GSCALER has relative error $< 10\%$, whereas the second best performing algorithm *UpSizeR* has the average relative error 40%

For *Epinions*, GSCALER is the most stable and accurate algorithm, with a consistent absolute error < 0.25. *RW* performs well for large s, but does badly for small s.

5.3.8 Scaling Up

To save space, Fig.17 plots performance of both *Slashdot* and *Epinions* for each graph property.

Similar to scaling down, GSCALER improves the accuracy of indegree/outdegree/bidegree distribution significantly. For effective diameter, GSCALER and SKG are the best performing algorithms. GSCALER is the best algorithm which produces the most accurate results for average clustering coefficient, largest SCC ratio, and average shortest path length.

5.3.9 Summary of Comparisons

For indegree, outdegree and bidegree distributions, GSCALER reduces the error from about 0.1 for the other algorithms to about 0.01. This is expected since GSCALER uses f_{bi} to construct $\widetilde{G}$, and agrees with the theoretical bound in Sec. 5.2.

GSCALER only uses $\mathcal{P}_{sub} = \{f_{bi}, f_{corr}\}$ to construct $\widetilde{G}$, but measures similarity to G with a larger set $\mathcal{P}$ of both local and global properties listed in Sec. 4.1. The results show that enforcing $\mathcal{P}_{sub}$ suffices to induce similarity for $\mathcal{P}$.

6. LIMITATION

Unlike the other algorithms, a user can choose both $\widetilde{n}$ and $\widetilde{m}$ for GSCALER. Table 5 illustrates GSCALER's accuracy for *Slashdot*, using different $\widetilde{n}$ and $\widetilde{m}$.

However, GSCALER cannot guarantee similarity for arbitrary choices of $\widetilde{n}$ and $\widetilde{m}$ — the error bound in *Theorem 4* becomes loose for large $|r|$. For example, if $n = 1000$ and $m = 5000$, but $\widetilde{n} = 2000$ and $\widetilde{m} = 500000$, one cannot expect to find any $\widetilde{G}$ similar to G.

There is a Densification Law [23] that says, if $G_1, G_2, G_3, \ldots,$ are snapshots of a growing graph, then

$$E(G_i) \propto V(G_i)^\alpha \quad \text{for some } 1 \le \alpha \le 2$$

If $\widetilde{G}$ follows such a law, then $(\widetilde{m}/m) = (\widetilde{n}/n)^\alpha$. If $\widetilde{n} = sn$ and $\widetilde{m} = (1+r)sm$, then

$$\frac{\widetilde{m}}{\widetilde{n}^\alpha} = \frac{(1+r)sm}{(sn)^\alpha} = \frac{m}{n^\alpha}\frac{1+r}{s^{(\alpha-1)}},$$

Therefore, for $\widetilde{G}$ to follow the Densification Law, the user must choose $r > 0$ for $s > 1$, and $r < 0$ for $s < 1$. In other words, $\widetilde{m} > m\widetilde{n}/n$ for $\widetilde{n}/n > 1$, and $\widetilde{m} < m\widetilde{n}/n$ for $\widetilde{n}/n < 1$.

Note that modeling the evolution of n and m is an interesting problem that is relevant, but orthogonal to GSP.

Slashdot		Effective Diameter	Largest SCC	ASPL	CC
$n = 77360$	$m = 828161$	5	0.909	3.74	0.052
$\widetilde{n} = 4421$	$\widetilde{m} = 32179$	5	0.916	3.59	0.058
$\widetilde{n} = 4421$	$\widetilde{m} = 33831$	5	0.916	3.52	0.053
$\widetilde{n} = 4421$	$\widetilde{m} = 34399$	5	0.916	3.62	0.058
$\widetilde{n} = 15472$	$\widetilde{m} = 141583$	5	0.916	3.57	0.053
$\widetilde{n} = 15472$	$\widetilde{m} = 144895$	5	0.916	3.51	0.060
$\widetilde{n} = 15472$	$\widetilde{m} = 147849$	5	0.916	3.45	0.059
$\widetilde{n} = 154720$	$\widetilde{m} = 1669903$	5	0.917	3.71	0.055
$\widetilde{n} = 154720$	$\widetilde{m} = 1671288$	5	0.917	3.71	0.062
$\widetilde{n} = 154720$	$\widetilde{m} = 1672057$	5	0.917	3.70	0.062
$\widetilde{n} = 386800$	$\widetilde{m} = 4182213$	5	0.917	3.89	0.058
$\widetilde{n} = 386800$	$\widetilde{m} = 4186353$	5	0.917	3.89	0.048
$\widetilde{n} = 386800$	$\widetilde{m} = 4190908$	5	0.917	3.89	0.053

Table 5: Gscaler accuracy for different $\widetilde{n}$ and $\widetilde{m}$.

7. CONCLUSION

We considered the problem of synthetically scaling a given graph. Our solution GSCALER first breaks G into pieces, scales them, then merges them using the degree and correlation functions from G.

Different from previous approaches, GSCALER gives user a choice for $\widetilde{n}$ and $\widetilde{m}$. We proved that GSCALER does not produce multiple edges between two nodes, and has a small distribution error even when the average degree of $\widetilde{G}$ differs from the original graph G.

Experiments with 2 well-known real datasets show that the $\widetilde{G}$ constructed by GSCALER is more similar to G for most properties than random walk, forest fire, *UpSizeR* and Stochastic Kronecker Graph.

Our current work aims to extend GSCALER to scale relational databases by representing the tables as graphs.

8. REFERENCES

[1] N. K. Ahmed, N. Duffield, et al. Graph sample and hold: A framework for big-graph analytics. In *Proc. KDD*, pages 1446–1455, 2014.

[2] N. K. Ahmed, J. Neville, and R. Kompella. Network sampling: From static to streaming graphs. *TKDD*, 8(2):7, 2014.

[3] Y.-Y. Ahn, S. Han, et al. Analysis of topological characteristics of huge online social networking services. In *Proc. WWW*, pages 835–844, 2007.

[4] W. Aiello, F. Chung, and L. Lu. A random graph model for power law graphs. *Experimental Mathematics*, 10(1):53–66, 2001.

[5] R. Albert and A.-L. Barabási. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74(1):47, 2002.

[6] A.-L. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999.

[7] D. Chakrabarti, Y. Zhan, and C. Faloutsos. R-mat: A recursive model for graph mining. In *SDM*, volume 4, pages 442–446. SIAM, 2004.

[8] S. Duan, A. Kementsietsidis, K. Srinivas, and O. Udrea. Apples and oranges: a comparison of rdf benchmarks and real rdf datasets. In *Proc. SIGMOD*, pages 145–156. ACM, 2011.

[9] P. L. Erdös and A. Rényi. On the evolution of random graphs. In *Publication of the Mathematical Institute of*

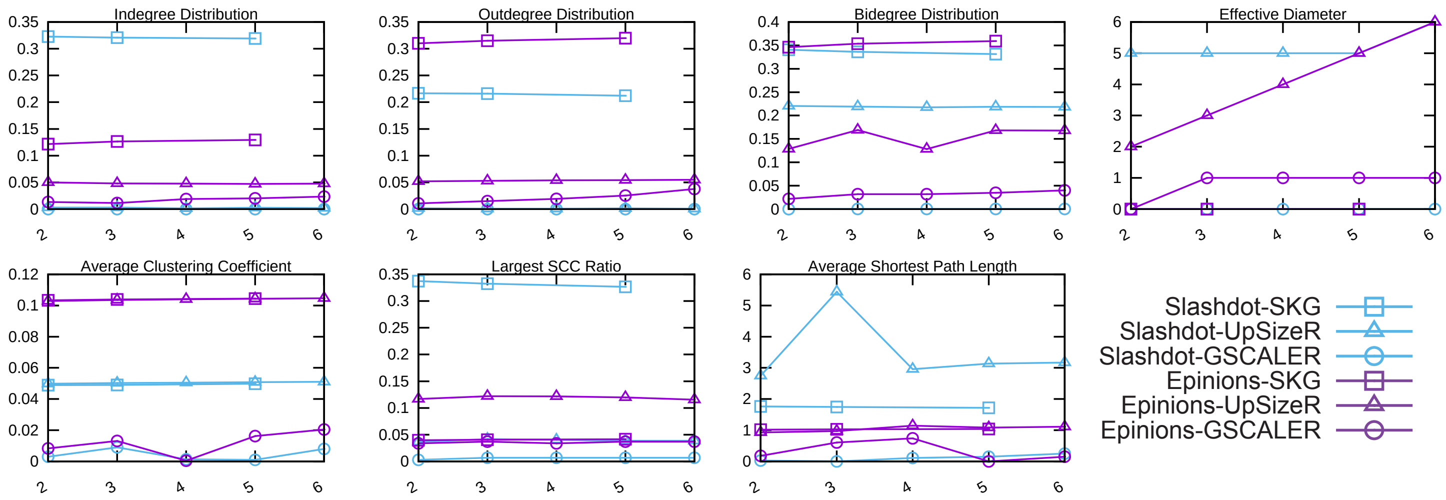


Figure 17: Scaling Up Experiments (s values on horizontal axes; legend format is dataset-algorithm.)

the Hungarian Academy of Science, pages 17–61, 1960.

[10] G. Fasano and A. Franceschini. A multidimensional version of the Kolmogorov-Smirnov test. Monthly Notices Royal Astronomical Society 255(1), 1987.

[11] A. D. Flaxman, A. M. Frieze, and J. Vera. A geometric preferential attachment model of networks ii. *Internet Mathematics*, 4(1):87–111, 2007.

[12] M. Gjoka, M. Kurant, et al. Walking in Facebook: A case study of unbiased sampling of OSNs. In *Proc. INFOCOM*, pages 2498–2506, 2010.

[13] L. A. Goodman. Snowball sampling. *The Annals of Mathematical Statistics*, pages 148–170, 1961.

[14] P. Hu and W. C. Lau. A survey and taxonomy of graph sampling. *CoRR*, abs/1308.5865, 2013.

[15] A. Kemper. *Valuation of Network Effects in Software Markets*. Physica, 2010.

[16] V. Krishnamurthy, M. Faloutsos, et al. Reducing large Internet topologies for faster simulations. In *Proc. NETWORKING*, pages 328–341, 2005.

[17] R. Kumar, P. Raghavan, et al. Extracting large-scale knowledge bases from the web. In *VLDB*, volume 99, pages 639–650, 1999.

[18] S. Lee, P. Kim, and H. Jeong. Statistical properties of sampled networks. *Physical Review E 73(1)*, 2006.

[19] J. Leskovec, D. Chakrabarti, et al. Realistic, mathematically tractable graph generation and evolution, using kronecker multiplication. In *Proc. PKDD 2005*, pages 133–145. Springer, 2005.

[20] J. Leskovec, D. Chakrabarti, et al. Kronecker graphs: An approach to modeling networks. *J. Machine Learning Research*, 11:985–1042, 2010.

[21] J. Leskovec and C. Faloutsos. Sampling from large graphs. In *Proc. KDD*, pages 631–636, 2006.

[22] J. Leskovec, D. Huttenlocher, and J. Kleinberg. Signed networks in social media. In *Proc. SIGCHI*, pages 1361–1370, 2010.

[23] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graphs over time: densification laws, shrinking diameters and possible explanations. In *KDD*, pages 177–187, 2005.

[24] J. Leskovec and A. Krevl. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, June 2014.

[25] J. Leskovec, K. J. Lang, et al. Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. *Internet Mathematics*, 6(1):29–123, 2009.

[26] R.-H. Li, J. Yu, L. Qin, R. Mao, and T. Jin. On random walk based graph sampling. In *ICDE*, pages 927–938, 2015.

[27] N. Metropolis, A. W. Rosenbluth, et al. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092, 1953.

[28] S. Mussmann, J. Moore, J. J. P. III, and J. Neville. Incorporating assortativity and degree dependence into scalable network models. In *Proc. AAAI*, 2015.

[29] S. Qiao and Z. M. Özsoyoğlu. Rbench: Application-specific rdf benchmarking. In *Proc. SIGMOD*, pages 1825–1838. ACM, 2015.

[30] B. Ribeiro and D. Towsley. Estimating and sampling graphs with multidimensional random walks. In *Proc. IMC*, pages 390–403, 2010.

[31] R. Staden. A strategy of dna sequencing employing computer programs. *Nucleic Acids Research*, 6(7):2601–2610, 1979.

[32] M. Stumpf, C. Wiuf, and R. May. Subnets of scale-free networks are not scale-free: Sampling properties of networks. *PNAS 102(12)*, pages 4221–4224, 2005.

[33] Y. C. Tay, B. T. Dai, D. T. Wang, E. Y. Sun, Y. Lin, and Y. Lin. Upsizer: Synthetically scaling an empirical relational database. *Inf. Syst.*, 38(8):1168–1183, 2013.

[34] D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):409–10.

[35] C. Wilson, B. Boe, et al. User interactions in social networks and their implications. In *Proc. EuroSys*, pages 205–218, 2009.