MomentSA: A Fast and Accurate Method for Stochastic Kronecker Graph Parameter Computing

Yongqiang Li, Zhiyuan Shao, Hong Huang, Yinuo Li, Hai Jin
Services Computing Technology and System Lab
Cluster and Grid Computing Lab
School of Computer Science and Technology
Huazhong University of Science and Technology, Wuhan, 430074, China
EMail: {liyongqiang, zyshao, honghuang, hjin}@hust.edu.cn

***Abstract*—Stochastic Kronecker Graph model is widely used to generate synthetic graphs that simulate real-world graphs. In this model, the initiator matrix decides the degree to which the synthetic graph approximates the real-world graph. The computing of initiator matrix, however, requires that the number of the nodes of input graph is the power of the dimension of the initiator matrix. In order to fulfill such requirement, some methods (e.g., KronFit and Moment) add isolated nodes to the input graph, which damages the input graph's properties. Other method (e.g., KronEM) predicts the links between the nodes after adding isolated nodes. Unfortunately, the prediction dramatically increases the complexity of computing.**

In this paper, we propose a method named as MomentSA to solve the problems. Our method completes the input graphs by leveraging the law of property changes of graphs, which does not need to predict the links as method KronEM. Simultaneously, our method uses the moment-based estimation and ADAM (*Adaptive Moment Estimation*) method to compute the initiator matrix, which can compute the initiator matrix quickly. Experiment results on our prototype implementation suggest that the initiator matrix computed from our method is more accurate than existing state-of-art systems, and the speed of computing is about three to four orders of magnitude faster than state-of-art systems.

***Index Terms*—Stochastic Kronecker Graph model; Initiator Matrix; Moment-based Estimation**

I. Introduction

Stochastic Kronecker Graph model is widely used in the synthesis of graphs [1, 2], data privacy protection, network complementary [3], and many other fields because of its structural characteristics[4] and rigorous mathematical theory support. A stochastic kronecker graph is created from the probability matrix P that is generated by the initiator matrix Θ through *Kronecker Multiplication* $\otimes$ and *Kronecker Power*[5]. As the coefficient of *Kronecker Power* is arbitrary positive integer, the synthetic graph G^* can be of arbitrary (large) size. This property of G^* is widely used to simulate real-world graphs: for example, evaluating the performance of graph processing systems when handling large graphs, predicting the properties of the real-world graphs during evolution, and many others.

In Stochastic Kronecker Graph model, one of the most important parameters is the initiator matrix, which decides the degree to which the synthetic graph approximates the real-world graph. In order to simulate a specific (say social or web) type of graphs, one needs to compute the initiator matrix from representative real-world graphs (we call them as *input graphs* in this paper) of that type. However, the computing requires the number of nodes (i.e., vertices) of the input graph to be the power of the dimension of initiator matrix (the best dimension is 2^1 according to[2]), so as to establish a one-to-one mapping between the nodes of the input graph and those of synthetic graph (rows of probability matrix P). Unfortunately, such requirement is unrealistic for the real-world input graphs.

Existing methods that solve above problems could be divided into two clusters: first, completing the input graph with isolated nodes, which have no edges at all; Second, predicting the links besides adding the isolated nodes according to the properties (e.g., distribution of edges) of the input graph. However, as the introduction of isolated nodes inevitably damages the properties (e.g., density and average degree) of the input graph, such method (employed by KronFit [2] and Moment [6]) generally leads to inaccurate[2] results. For the second method (employed by KronEM [3]), although it alleviates the problem by predicting the links between the nodes after adding the isolated nodes, it introduces high computational overheads and thus inapplicable for large input graphs. As Figure 1 shown, Figure 1(a) shows the first method to complete graph, which only adds isolated nodes without adding edges. Figure 1(b) shows the second method that filling the edges of graph Z by input graph G. However, this method needs to infer the edges of the graph.

How to complete the input graphs, without damaging the properties of the input graph and avoiding the high complexity of predicting the links, becomes a huge challenge in mining the initiator matrix. In this paper, we propose a method named as MomentSA to solve above problems. Our method completes the input graphs by leveraging the law of property changes of graphs, which does not need to predict the links as method KronEM. Simultaneously, our method uses the moment-based estimation and ADAM (*Adaptive Moment Estimation*) method to compute the initiator matrix, which can compute the initiator matrix quickly. The contributions of this paper are listed in the following:

[1]We regard the dimension of initiator matrix as 2 in following discussions.
[2]We say an initiator matrix is more "accurate", if the properties (e.g., number of edges, vertices, and other shapes) of the synthetic graph that is generated by such matrix are closer to the input graph.

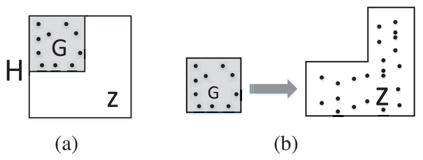


Figure 1: H represents the complete graph whose size equals the power of the dimension of the initiator matrix. G represents the input graph, and Z represents isolated nodes part. (a) in the first method, H is completed by isolated nodes part Z, which have no edges at all. (b) in the second method, H is completed by the Z, which edges are inferred by the input graph G.

- discloses the fact that for a series of sub-graphs generated by uniformly sampling nodes from an input graph, the number of specific shapes (e.g., edges, hairpins, tripins, and triangles) are proportional to the power of the number of nodes in any of such sub-graph, and the exponents are constant.
- proposes the MomentSA, which leverages above law to complete the input graph, and uses moment-based estimation and ADAM method to compute the initiator matrix.
- implements a prototype system that realizes MomentSA, and extensively evaluates our prototype implementation.

The rest of this paper is organized as follows: Section II introduces related work. Section III introduces the background. Section IV gives the algorithm of MomentSA. Section V evaluates our prototype implementation and compares MomentSA with other methods. Section VI concludes the paper.

II. Related Work

Graph models are used to generate synthetic graphs, which can be used for simulation studies. Graph models can be broadly divided into five categories [7]: Random graph models, Preferential attachment models, Optimization-based models, Tensor-based models [1, 8, 9], and Generators for specific graphs. Random graph models are used to generate graphs by a random process, such as Erdös-Rényi model, Power-law Random graph. Preferential attachment models [10] make the "rich" get "richer" and it includes Forest Fire model, Butterfly model, etc. Optimization-based models consider the *resource optimizations* and it includes Small World model, the Waxman model, etc. Generators for specific graphs are order to generate some special-purpose structure graphs, such as Internet-specific model. However, almost all graph models focus on only one or two patterns (e.g., Power Law Distribution, Small Diameters, Densification Power Law [11]). There is a need for models which can match most of the graph patterns. Tensor-based models are a step in this direction, whcih are based on *self-similar* mechanisms and it includes R-MAT[8] model and Stochastic Kronecker Graph model. Many works [1, 2] show that Stochastic Kronecker Graph model can match most of the graph patterns, such as multinomial degree distributions, static diameter/effective diameter, multinomial distributions of eigenvalues, and community structure.

The methods used to mine the initiator matrix in Stochastic Kronecker Graph model can be classified into two categories: *Maximum Likelihood Estimation* (MLE) and *Moment Based Estimation* (MBE). MLE (used in KronFit [2] and KronEM [3]) finds the initiator matrix Θ to map real-world graph G with the maximum likelihood $P(G)$. MBE (used in Moment [6]) computes the initiator matrix by minimizing the distance between the observed values of graph's structure attributes (e.g., edges, hairpins (2-stars or wedges), tripins (3-stars), and triangles) with those of the expected values of these structure attributes. However, existing methods require the number of the nodes of input graph to be power of the dimension (i.e., 2) of the initiator matrix. Therefore, it is necessary to complete the input graph before mining the initiator matrix. Existing methods of completing the input graphs could be divided into two clusters: some of existing methods (e.g., KronFit and Moment) simply add isolated nodes to the input graph, which damages the input graph's properties. Other methods (e.g., KronEM) amend this problem by predicting the links between the nodes after adding isolated nodes. Such methods, however, dramatically increase the complexity of computing.

III. Background

The methods used to compute the initiator matrix can be classified into two categories: Maximum Likelihood Estimation and Moment Based Estimation.

• ***Maximum Likelihood Estimation*** **(MLE)** MLE finds the initiator matrix Θ to map real-world graph G with the maximum likelihood $P(G)$ [2]. Because there is no special meaning of node's label, the likelihood has contributions from $N!$ permutations σ. That is $P(G) = \sum_{\sigma} P(G|\Theta, \sigma) P(\sigma|\Theta)$. Then the objective function for MLE is:

$$\arg\max_{\Theta} \sum_{\sigma} P(G|\Theta, \sigma) P(\sigma|\Theta)$$

For every likelihood of each permutation σ, it needs to traverse all elements of probability matrix P with complexity $O(N^2)$. Therefore, the complexity of such method is $O(N!N^2)$ at all. The representative systems that employs MLE are KronFit[2] and KronEM [3].

• ***Moment Based Estimation*** **(MBE)** MBE computes the initiator matrix by minimizing the distance between the observed values of graph's structure attributes (e.g., edges(E), hairpins(2-stars or wedges)(H), tripins(3-stars)(T), and triangle(Δ)) with those of the expected values of these structure attributes. For initiator matrix $\Theta = \begin{bmatrix} a & b \\ b & c \end{bmatrix}$, the objective function is as follows :

$$\min_{a,b,c} = \sum_{F} \frac{(F - \mathbb{E}_{a,b,c}(F))^2}{\mathbb{E}_{a,b,c}(F)}$$

where the summation is the attributes $F \in \{E,\ \Delta,\ H,\ T\}$ and the minimization is taken over $0 \leq c \leq a \leq 1$ and $0 \leq b \leq 1$.The complexity of MBE mainly resides in computing edges ($O(E)$), hairpins ($O(E)$), tripins ($O(E)$), and triangles ($O(E^{1.5})$[12]). Therefore, time complexity of MBE is $O(E^{1.5})$. MBE is computationally much simpler and lighter than MLE, but it can not get the mapping between probability matrix P and the real-world graphs.

IV. Proposed Method

We propose a novel method named MomentSA to compute initiator matrix. This method includes two parts: graph completion and parameter computing.

A. Graph Completion

The objective of graph completion is to extend the input graph, such that the number of nodes of the expanded graph becomes power of 2. At the same time, the expanded graph has similar properties as the input graph. In this subsection, we will introduce the theorem used to expanding the input graph after a definition on a sampling method.

Definition 1. ***Uniform Random Node Sampling (URNS) [13]:*** *Given an input graph $G = \{V, E\}$, where $V = \{v_1, v_2, \ldots, v_n\}$ is the set of nodes, and $E = \{e_1, e_2, \ldots, e_n\}$ is the set of edges, an URNS subgraph $G^{'}$ is consisted of nodes that are randomly selected with uniform probability, as well as the edges connecting the selected nodes.*

As illustrated in Figure 2, an URNS subgraph $G^{'}$ is generated by selecting the nodes from the original graph G with equal probability, edges that connect the chosen nodes are then added to subgraph $G^{'}$.

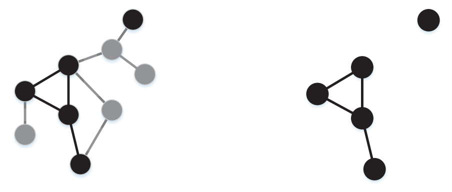

(a) Original graph (b) Subgraph $G^{'}$

Figure 2: Uniform Random Node Sampling

Considering a series of subgraphs obtained by URNS on an input graph, we have following theorem.

Theorem 1. ***(DPL for URNS subgraphs)*** *Let $\widetilde{G}$ be a series of subgraphs formed by URNS from an undirected graph G, and $\widetilde{G} = \{G(1), G(2), \ldots, G(K)\}$, where K is a positive integer bigger than 1. For any subgraph $G(t)$, where $t \in \{1, 2, \ldots, K\}$, taken from $\widetilde{G}$, denote the number of the edges, tripins, hairpins, triangles, and nodes of $G(t)$ as $E(t), T(t), H(t), \Delta(t), N(t)$ respectively, the relationships between $E(t), T(t), H(t), \Delta(t)$, and $N(t)$ are: $E(t) \propto N(t)^{\alpha}, T(t) \propto N(t)^{\beta}, H(t) \propto N(t)^{\gamma}, \Delta(t) \propto N(t)^{\delta}$, and the exponent coefficients are $\alpha = 2, \beta = 4, \gamma = 3, \delta = 3$.*

Proof. Denote the number of nodes and edges of the input undirected graph G as N and E respectively. Use $A(G)$ to denote the adjacency matrix of G, whose elements are $A_{ij} \in \{0, 1\}$, and $i, j \in \{0, \ldots, N-1\}$. As the graph is undirected, we have: $E = \frac{1}{2}\sum_{ij} A_{ij}$.

For any subgraph $G(s)$ taken from $\widetilde{G}$, according to the properties of URNS, the probability used during generating $G(s)$ is $\frac{N(s)}{N}$. Consider an edge e_{ij}, it exists in $G(s)$ only when both its endpoint nodes are chosen and appear in $G(s)$. The probability for $G(s)$ to have e_{ij} is thus $\left(\frac{N(s)}{N}\right)^2 \cdot A_{ij}$. We now can compute the number of edges in $G(s)$:

$$E(s) = \frac{1}{2}\sum_{ij}\left(\left(\frac{N(s)}{N}\right)^2 A_{ij}\right) = \left(\frac{N(s)}{N}\right)^2 E$$

Then, take another subgraphs $G(t)$ from $\widetilde{G}$, the number of edges in $G(t)$ is:

$$E(t) = \left(\frac{N(t)}{N}\right)^2 E$$

Comparing these two subgraphs, we have:

$$\frac{logE(s) - logE(t)}{logN(s) - logN(t)} = \frac{log\left(N(s)/N(t)\right)^2}{log\left(N(s)/N(t)\right)} = 2$$

Therefore, $E(t) \propto N(t)^{\alpha}$ and $\alpha = 2$.

Similarly, the number of tripins in $G(s)$ can be computed by using following equation:

$$T(s) = \frac{1}{6}\sum_{ijkl}\left(\left(\frac{N(s)}{N}\right)^4 A_{ij}A_{ik}A_{il}\right) = \left(\frac{N(s)}{N}\right)^4 T$$

and that in $G(t)$ is:

$$T(t) = \frac{1}{6}\sum_{ijkl}\left(\frac{N(t)}{N}\right)^4 A_{ij}A_{ik}A_{il} = \left(\frac{N(t)}{N}\right)^4 T$$

then

$$\frac{logT(s) - logT(t)}{logN(s) - logN(t)} = \frac{log\left(N(s)/N(t)\right)^4}{log\left(N(s)/N(t)\right)} = 4$$

Hence, we have $T(t) \propto N(t)^{\beta}$, where t is arbitrarily chosen among 1 to K, and the exponent coefficient β is 4.

Similarly, as the hairpins and triangles are $H = \frac{1}{2}\sum_{ijk}(A_{ij}A_{ik})$ and $\Delta = \frac{1}{6}\sum_{ijk}(A_{ij}A_{ik}A_{jk})$, we can infer $\frac{logH(s) - logH(t)}{logN(s) - logN(t)} = 3$ and $\frac{log\Delta(s) - log\Delta(t)}{logN(s) - logN(t)} = 3$. Therefore, we have $H(t) \propto N(t)^{\gamma}, \Delta(t) \propto N(t)^{\delta}, \gamma = 3, \delta = 3$. $\square$

Theorem 1 has many similarities as the *Densification Power Law* (DPL) [11] proposed in 2005. DPL discloses the fact that the number of edges is proportional to power of the number of nodes in any snapshot from an evolving graph. Such regularity has been proven in social graphs[14], information graphs, and many others. On the other hand, Theorem 1 also states that the numbers of specific shapes (i.e., tripins, hairpins, triangles besides edges) are proportional to power of the number of nodes for any subgraph obtained from URNS for any input graph. Due to such similarities, we name Theorem 1 as “DPL for URNS subgraphs”.

The exponent coefficients in DPL, although approximately remain constant for one specific (say Twitter or Youtube, alone) graph, are different when comparing different graphs (e.g., comparing Twitter and Youtube graph). At this point, Theorem 1 is different from the DPL, and states that the exponent coefficients for a series of subgraphs obtained by URNS from *any* input graph are constants. In order to validate such facts, we will conduct experiments to further prove Theorem 1 in subsection V-A.

MomentSA leverages Theorem 1 to complete graph. First, the number of edges, tripins, hairpins, and triangles (denote as E, T, H, and Δ respectively) in an input graph G are obtained by graph mining. Second, G is expended to $G^{\star}$ by using Theorem 1 such that the number of nodes in $G^{\star}$ becomes power of 2. Last, the number of edges, hairpins, tripins and triangles (denote as $E^{'}, T^{'}, H^{'}, \Delta^{'}$ respectively) in $G^{\star}$ are computed according to Theorem 1. For example, given an input graph G, which has 5 nodes, 7 edges, 14 hairpins, 5 tripins, and 3 triangles, after graph completion, $G^{\star}$ will have $2^{\lceil log_2^5 \rceil} = 8$ nodes, $E^{'} = \left(\frac{8}{5}\right)^2 \times 7 = 18$ edges, $H^{'} = \left(\frac{8}{5}\right)^3 \times 14 = 57$ hairpins, $T^{'} = \left(\frac{8}{5}\right)^4 \times 5 = 33$ tripins, and $\Delta^{'} = \left(\frac{8}{5}\right)^3 \times 3 = 12$ triangles. By such method, MomentSA estimates the properties' values of the completed graph in the constant time and does not need to predict the edges connecting the newly-added nodes as in KronEM.

B. Computing Parameters

MomentSA uses ADAM algorithm[15], which is a combination of the Momentum, AdaGrad [16], and RMSProp [17] algorithm, to compute the initiator matrix. Momentum adds the influence of momentum factors, such that the step-size for each gradient descent is affected not only by the current gradient but also by the accumulation of historical gradients. For example, as shown in Figure 3, a ball is affected by momentum during the sliding, and it does not stay at point A, but moves to further right. AdaGrad and RMSProp algorithm mainly change the range of the step-size, such that step-size is in a certain range. ADMA can obtain the optimal solution of parameters by combining above algorithms.

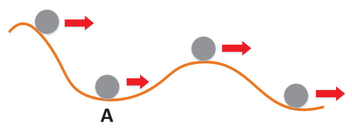


Figure 3: Diagram of the momentum algorithm

Algorithm 1 lists the whole process of MomentSA. It mainly includes four steps: first, computing the number of edges, tripins, hairpins, and triangles in the input graph G (line 4), which are denoted as E, T, H, and Δ. Second, expanding G to $G^{\star}$ and estimating the number of edges, hairpins, tripins, and triangles in $G^{\star}$ which is $E^{'}, T^{'}, H^{'}, \Delta^{'}$ by using Theorem 1 (line 6 to 11). Third, using MBE to construct following objective function:

$$\arg\min_{\Theta} = \min\left\{\frac{1}{2}\left(\sum\left(\frac{F^{'} - F}{F}\right)^2\right)\right\} \quad (1)$$

where $F \in \{E^{'}, T^{'}, H^{'}, \Delta^{'}\}$ and $F^{'} \in \{\mathbb{E}(E), \mathbb{E}(\Delta), \mathbb{E}(H), \mathbb{E}(T)\}$. Finally, ADMA algorithm is used to compute the parameters of the initiator matrix (line 12).

V. Evalution

In this section, we first conduct experiments to validate Theorem 1, and then evaluate the accuracy and speed of MomentSA on several synthetic and real-world graphs. MomentSA is implemented on top of SNAP [18] (version c++ 4.0). The datasets used in experiments are listed in Table I. All graphs are undirected and all edges are unweighted. SynGraph1 and SynGraph2 are synthetic graphs generated with the $\Theta_1 = [0.85\ 0.55; 0.55\ 0.4]$, $\Theta_2 = [0.99\ 0.5; 0.5\ 0.6]$, and $k = 14$.

Algorithm 1 The objective function of MomentSA is $f(\Theta) = \arg\min_{\Theta} = \min\left\{\frac{1}{2}\left(\sum\left(\frac{F^{'}-F}{F}\right)^2\right)\right\}$.

```
1: Input: graph $G$ on $N$ nodes
2: Output: initiator matrix $\Theta\,(N_0 \times N_0)$
3: $\Theta_0$ (Initial parameter matrix)
4: $k = \lceil log_{N_0}^{N} \rceil$
5: Compute : $E(Edge),\ T(Tripin),\ H(Hairpin),\ \Delta(Triangle)$
6: if $N$ equals $N_0^k$ then
7:     $E^{'} = E,\ T^{'} = T,\ H^{'} = H,\ \Delta^{'} = \Delta$
8: else
9:     $E^{'} = \left(\frac{N_0^k}{N}\right)^2 E,\ T^{'} = \left(\frac{N_0^k}{N}\right)^4 T$
10:    $H^{'} = \left(\frac{N_0^k}{N}\right)^3 H,\ \Delta^{'} = \left(\frac{N_0^k}{N}\right)^3 \Delta$
11: end if
12: $\Theta$ = ADAM($\Theta_0$ , $f(\Theta)$ )
13: return $\Theta$
```

Table I: DATASET OF REAL-WORLD GRAPHS AND SYNTHETC GRAPHS

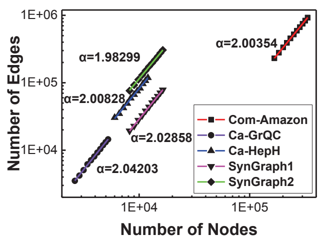
Dataset	Nodes	Edges
Ca-GrQc	5242	14484
Ca-HepTh	12008	118489
Com-Amazon	334863	925872
Email-Enronun	36692	183831
Soc-Epinions1	75879	405740
SynGraph1 ([0.85 0.55;0.55 0.4])	16384	78329
SynGraph2 ([0.99 0.5;0.5 0.6])	16384	305609

A. Validation on Theorem 1

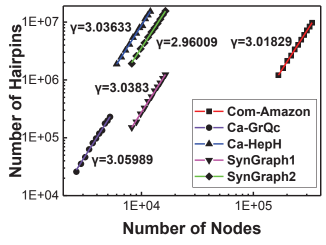
We validate Theorem 1 through the following experiments. First, we generate a series of sub-graphs by using URNS with uniform probability from 0.95 to 0.5 (the step size is 0.05), and then measure the variation of edges, hairpins, tripins, and triangles as well as the number of nodes of these subgraphs. The results are shown in Figure 4. Figure 4(a) describes the relationship between edges and nodes of graph in logarithmic scale. It can be observed that $E(t) \propto N(t)^{\alpha}$ holds, and the exponent coefficients α are 2.00354, 2.04203, 2.0082, 2.02858, and 1.98299 respectively. Similarly, from Figure 4(b), 4(c), and 4(d), we can observe that $T(t) \propto N(t)^{\beta}, H(t) \propto N(t)^{\gamma}, \Delta(t) \propto N(t)^{\delta}$ also hold, and the slopes, which denote the exponent coefficients of β, γ, and δ, are very close to 3, 4, and 3 respectively. These results validate Theorem 1 experimentally for both real-world and synthetic graphs.

B. Experiments on Synthetic Graphs

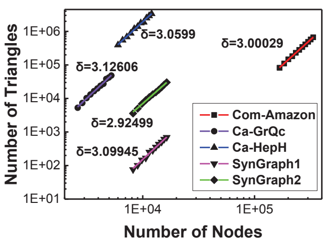
This experiments on synthetic graphs are conducted on a series of subgraphs $\widetilde{G}$ obtained by URNS on SynGraph1 as in subsection V-A. The objective of these experiments



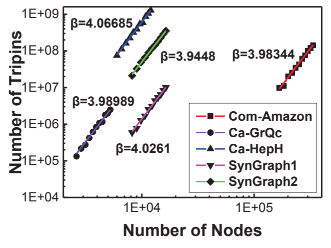
(a) Edge



(b) Hairpin



(c) Triangle



(d) Tripin

Figure 4: The number of edges, hairpins, triangles, tripins and that of the nodes in the URNS subgraphs (in log-log scale)

is to recover the initiator matrix by using these subgraphs. Subsection V-B1 reports the closeness (i.e., accuracy) of the recovered initiator matrix and the original one, and subsection V-B2 reports the speed of computing.

1) Recover the Initiator Matrix: In order to measure the accuracy of results, we defined the distance metric, D_{SUMABS}, to measure the distance from the recovered initiator matrix $\hat{\Theta}$ and the original initiator matrix (i.e., Θ = [0.85 0.55;0.55 0.4]) of SynGraph1. D_{SUMABS} is computed by using: $D_{SUMABS} = \left| \left| \hat{\Theta} \right| - |\Theta| \right|$, where $|\Theta| = \sum_{i,j} \Theta_{ij}$.

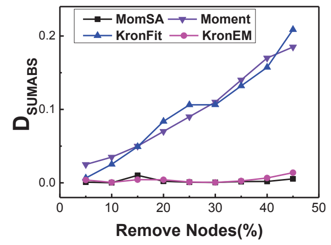
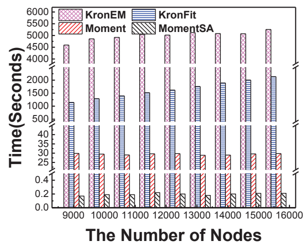


Figure 5: Moment, MomentSA, KronFit, and KronEM recover the initiator matrix D_{SUMABS}

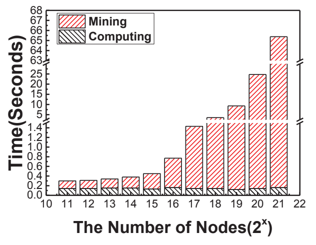
The results are shown in Figure 5. From this figure, we can observe that with more nodes removed from SynGraph1, the computed results from KronFit and Moment are increasingly deviating from original initiator matrix. The reason behind such results is that these two methods complete the input graph by adding isolated nodes, which damages the properties of the input graph. On the other hand, the computed results from both KronEM and MomentSA are close to the original initiator matrix, even when about half of the nodes are removed from the original graph.

2) Speed: Figure 6(a) compares the total execution time of all methods. It can be observed that MomentSA value is the fastest, and it is about three to four orders of magnitude faster than KronEM and KronFit. The main reason is that MomentSA uses the MBE to compute initiator matrix, which does not need to traverse the graph many times as MLE. Meanwhile, MomentSA employs the ADAM method, which is about two orders of magnitude faster than the grid search method used by Moment method.

As the execution time of MomentSA is consisted of two parts: mining the graph's features (edges, hairpins, tripins, and triangles, etc.) and the computing the parameters, Figure 6(b) shows the breakdown of the execution time of MomentSA. We can observe that graph mining actually dominates the whole computing process in MomentSA. The computing part, however, remains small and constant during the computation.



(a) Total execution time



(b) MomentSA exec. time breakdown

Figure 6: Comparison on execution times

C. Experiments on Real-World Graphs

We choose five graphs, CA-HepPh, CA-GrQc, Com-Amazon, Email-Enronun, and Soc-Epinions1, from Table I to evaluate the effectiveness of MomentSA and exiting methods on real-world graphs. The experiments are conducted by first mining the input graphs to obtain their initiator matrices, and then use the Krongen tool from SNAP suite to generate synthetic graph. As the number of nodes of the synthetic graphs generated by using Krongen has to be power of 2, we scale down the synthetic graphs by using our Theorem 1 such that the synthetic graphs have the same amount of nodes as original real-world graphs. Table II compares several properties (i.e., number of shapes, density) of the synthetic graphs obtained by different methods and those of their corresponding real-world graphs. The digits in OBJ column are computed by using Equation 1, and they indicate the closeness between the synthetic graphs and the real-world graphs. Smaller OBJ values mean they are more close. We can observe from Table II that compared with those obtained by other methods, the synthetic graphs generated by using MomentSA are the closest to the original real-world graphs. This highlights the accuracy of MomentSA on simulated real-world graphs.

VI. Conclusion and Future Works

In this paper, we propose MomentSA, a fast and accurate method for stochastic Kronecker graph parameter computing. This method leverages the DPL-like law on the uniformly sampled graphs to complete the input graph, so as to maintain

Table II: EXPERIMENTS ON REAL-WORLD GRAPHS (OBJ COLUMN: SMALLER IS BETTER)

Dataset	Method	Nodes	Edges	Hairpins	Tripins	Triangles	Density	OBJ
Ca-HepPH	Source	12008	118489	15278011	1.28E+09	3358499	0.001644	-
	MomentSA	12008	130646	13209804	1.35E+09	73457	0.001812	0.4943
	KronEM	12008	104166	5297721	2.11E+08	19051	0.001445	1.0633
	Moment	12008	69191	4876945	3.9E+08	21241	0.00096	1.0519
	KronFit	12008	57351	2258129	8.4E+07	7308	0.000796	1.4301
Ca-GrQc	Source	5242	14484	229867	2482738	48260	0.001054	-
	MomentSA	5242	14777	225362	2526556	670	0.00107	0.4868
	KronEM	5242	13613	142917	910185	159	0.00099	0.7707
	Moment	5242	6137	55071	441030	137	0.000447	1.2905
	KronFit	5242	4400	10339	110001	13	0.00032	1.6548
Com-Amazon	Source	334863	925872	9752186	1.43E+08	667129	0.000017	-
	MomentSA	334863	777073	11061164	1.34E+08	476	0.000014	0.5229
	KronEM	334863	964143	8953293	4.31E+07	100	0.000017	0.7475
	Moment	334863	309209	2315082	1.77E+07	288	0.000006	1.3955
	KronFit	334863	411110	2146203	7.29E+07	15	0.000007	1.4088
Email-Enronun	Source	36692	183831	25566893	3.48E+09	717044	0.000273	-
	MomentSA	36692	205993	22707620	3.7E+09	51776	0.000306	0.4460
	KronEM	36692	181375	10111984	6.57E+08	17594	0.000269	0.9873
	Moment	36692	65255	3683171	3.59E+08	5825	0.000097	1.4683
	KronFit	36692	50046	1412382	6.009E+07	1793	0.000074	1.6915
Soc-Epinions1	Source	75879	405740	74201120	8.13E+09	1624481	0.000141	-
	MomentSA	75879	472658	53300946	8.94E+09	80599	0.000164	0.5098
	KronEM	75879	369924	24569079	2.14E+09	28498	0.000129	0.9816
	Moment	75879	156628	9295259	9.73E+08	10021	0.000054	1.4524
	KronFit	75879	112178	4154328	2.59E+08	3721	0.000039	1.6737

the input graph's properties during completion. MomentSA does not need to predict the links among the added nodes, and thus avoids the high computing overheads for link prediction as in KronEM. Experiments on synthetic graphs show that MomentSA can faithfully recover the initiator matrix even when about half of the nodes of a synthetic graph is missing. Experiments on real-world graphs show that MomentSA can generate synthetic graphs that are very close to the input real-world graph. Moreover, the computing speed of MomentSA is about three to four orders of magnitude than existing state-of-art systems. Currently, MomentSA works only for undirected graphs, we plan to extend it to support directed graphs in the future.

Acknowledgment

This work is supported by the Fundamental Research Funds for the Central Universities of China under Grant No.2017KFYXJJ066.

References

[1] S. I. Moreno, J. Neville, and S. Kirshner, "Learning mixed kronecker product graph models with simulated method of moments," in *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2013, pp. 1052–1060.

[2] J. Leskovec, D. Chakrabarti, J. Kleinberg, C. Faloutsos, and Z. Ghahramani, "Kronecker graphs: An approach to modeling networks," *Journal of Machine Learning Research*, vol. 11, no. 2, pp. 985–1042, 2010.

[3] M. Kim and J. Leskovec, "The network completion problem: Inferring missing nodes and edges in networks," in *Proceedings of the 2011 SIAM International Conference on Data Mining*, 2011, pp. 47–58.

[4] A. J. Chin, T. D. Goodrich, M. P. OBrien, F. Reidl, B. D. Sullivan, and A. van der Poel, "Asymptotic analysis of equivalences and core-structures in kronecker-style graph models," in *Data Mining (ICDM), 2016 IEEE 16th International Conference on*, 2016, pp. 829–834.

[5] J. Leskovec and C. Faloutsos, "Scalable modeling of real graphs using kronecker multiplication," in *Proceedings of the 24th international conference on Machine learning*, 2007, pp. 497–504.

[6] D. F. Gleich and A. B. Owen, "Moment-based estimation of stochastic kronecker graph parameters," *Internet Mathematics*, vol. 8, no. 3, pp. 232–256, 2012.

[7] D. Chakrabarti, C. Faloutsos, and M. McGlohon, *Graph Mining: Laws and Generators*, ser. Advances in Database Systems. Springer, 2010, vol. 40, pp. 69–123.

[8] D. Chakrabarti, Y. Zhan, and C. Faloutsos, "R-mat: A recursive model for graph mining," in *Proceedings of the 2004 SIAM International Conference on Data Mining*, 2004, pp. 442–446.

[9] H. Park and M. Kim, "Trilliong: A trillion-scale synthetic graph generator using a recursive vector model," in *Proceedings of the 2017 ACM International Conference on Management of Data*, 2017, pp. 913–928.

[10] G. Even, R. Levi, M. Medina, and A. Rosén, "Sublinear random access generators for preferential attachment graphs," in *Proceedings of 44th International Colloquium on Automata, Languages, and Programming*, 2017, pp. 1–15.

[11] J. Leskovec, J. Kleinberg, and C. Faloutsos, "Graphs over time: densification laws, shrinking diameters and possible explanations," in *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, 2005, pp. 177–187.

[12] T. Schank, D. Wagner, T. Schank, D. Wagner, D. Watts, S. Strogatz, L. Laura, S. Leonardi, S. Millozzi, U. Meyer *et al.*, "Finding, counting and listing all triangles in large graphs," in *4th International Workshop on Experimental and Efficient*, vol. 2461, 2005, pp. 698–710.

[13] J. Leskovec and C. Faloutsos, "Sampling from large graphs," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2006, pp. 631–636.

[14] C. Zang, P. Cui, and C. Faloutsos, "Beyond sigmoids: The nettide model for social network growth, and its applications," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 2015–2024.

[15] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014. [Online]. Available: http://arxiv.org/abs/1412.6980

[16] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, no. 7, pp. 2121–2159, 2011.

[17] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop, coursera: Neural networks for machine learning," *University of Toronto, Tech. Rep*, 2012.

[18] J. Leskovec and R. Sosič, "Snap: A general-purpose network analysis and graph-mining library," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 8, no. 1, pp. 1–20, 2016.