Inference of *Kronecker* Structure

Myunghwan Kim
Stanford University
California, USA
mykim@stanford.edu

ABSTRACT
A stochast Kronecker graph presents a generative model for the social and information networks. By means of EM algorithm with various version of MCMC samplings,we suggest a way to catch the potential Kronecker structure. This model allows us to infer how the latent part of graph is organized as well as to predict how the current graph evolves over time.

General Terms
Kronecker Graph

Keywords
Kronecker, EM, Graph Prediction

1. INTRODUCTION
The synthetic graph model is essential for many reasons; for a simple example, it enables to simulate the graph and predict how it is going to grow. Inversely, in order to reduce the size without touching the key features, it can be used summarize or sample the graph.

In history, a lot of attempts to incorporate a probablistic graph model have been made. Among them, a stochastic Kronecker graph supplies abundant network features such as power-law degree distribution, even though it is controlled by a small number of paramters[2]. Furthermore, it gives nice intuition that nodes sharing common attributes show similar tendency of createing connections.

More than all, the Kronecker graph fulfills the purpose of synthetic graph; it summarizes the graph with a few paramters, and they are quite stable over time[2]. Because of these benifits, it is important to find the Kronecker parameters from the given graph.

We basically propose the Kronecker-fitting algorithm; however, the parameter estimation is just the tip of the iceberg. It can not only help us recover the randomly removed part of graph but also anticipate the graph structure in the future.

In section 2, we clearly define the problem and the approach of the solution follows it. Section 3 describes what we have tried and achieved where the fitting algorithm is provided. Finally, section 4 demonstrates the superiority of our approach compared to the previous one.

2. KRONECKER ESTIMATION

2.1 Problem Definition
The fundamental goal of this paper is to seek for the Kronecker structure in the graph. To state the problem mathematically, we want to find the Kronecker initiator $\Theta \in \mathbb{R}^{N_0 \times N_0}$ for the observed Graph G such that

$$\Theta_{max} = \arg\max_{\Theta} P(G; \Theta)$$

where $P(G; \Theta)$ is the likelihood of G parameterized by Θ. [2] has the same objective function; however, the Kronecker structure is defined only when the number of nodes is equal to N_0^k for an interger k. Unfortunately, it is impossible for every graph to satisfy this condition. To overcome this restriction, [2] appends isolated nodes until the number of nodes reaches a power of N_0, what we propose here is to introduce the latent subgraph the nodes of which are disjoint with G in the Kronecker graph. That is, we assume that G is the observed part in the latent Kronecker structure. Therefore, what we hope to maximize is the likelihood of G over the latent part of the graph, Z, as well as unknown permutation σ.

To distinguish our approach from the original one, we name each as KronEMFit and KronZeroFit, respectively.

2.2 Monte Carlo Expectation Maximization
A general approach for maximization over latent variables is the expectation-maximization(EM) algorithm, but the plain EM algorithm would not work well because we cannot derive the closed form of the posterior distribution. Instead, Monte Carlo EM(MCEM) algorithm[6] might be a good approach for our problem that sampling from the posterior is tractable. The skeleton of the EM algorithm is described in Algorithm A.1 and A.2.

2.3 Sampling from Posterior Distribution
However, it is definitely difficult to sample directly from the posterior $P(Z, \sigma | G, \Theta)$. Fortunately, the block Gibbs

sampling method is appropriate for this problem because sampling from the conditional posteriors, $P(Z|G,\sigma,\Theta)$ and $P(\sigma|G,Z,\Theta)$, is not difficult by following the procedure already defined in [2]. The former sampling is a kind of Kronecker-generating process, while the latter one is a Markov Chain Monte Carlo(MCMC) algorithm, in the form of Metropolis algorithm in [2]. Moreover, since the posterior distribution for permutations, $P(\sigma|G,Z,\Theta)$, is unchanged, we are able to exploit the same proposal process for MCMC. In other words, what we should focus on is the sampling algorithm from $P(Z|G,\sigma,\Theta)$.

The straightforward way is to determine the connectivity of each latent edge from a coin-toss, for the probability of each edge is independent of the other edges conditioned on σ and Θ. As written in [2], it is intractable because $O(|Z|)$ time is required and it can be as large as $O(N^2)$. Therefore, we need to make a detour to sample them using the fast generation of the Kronecker graph. Where sampling from $P(G,Z|\sigma,\Theta)$ is trivially achievable, one natural way to sample from $P(Z|G,\sigma,\Theta)$ is to use the acceptance-rejection algorithm. Briefly speaking, we pick edges up according to the fast Kronecker-generating algorithm as if there were no distinction between latent and observed parts of the graph. Then, we throw away the edges in case that they belong to the observed part. We can repeat this procedure until obtaining the expected number of edges.

2.3.1 Metropolized Gibbs Sampling

In the previous section, we present tractable algorithms, but they are not efficient enough. First, the Metropolis algorithm for permutation requires a proper number of of warming-up iterations. The number of those steps will undoubtedly increases according to the size of graph. Besides, as mentioned above, it takes $O(|E|)$ time to assign the latent variable, Z. The efficiency issue arises because those time-consuming jobs result in only one Gibbs sample. When we repeat this until achieving enough samples, the total running time is expected to be very huge. Unless this problem is resolved, the overall algorithm cannot finish in feasible time.

As MCMC provides a way to search for Monte Carlo samples in high dimensions, Metropolized Gibbs Sampling[4] can be a hint for this problem. When sampling each Gibbs block, corresponding Metropolis-Hastings algorithm[1] is respectively applied. In other words, Z and σ vary little by little for every Gibbs iteration in the proposal transition. The basic algorithm for σ does not need to change; however, no more warming-up is required between two consecutive Gibbs samples.

In the other hand, a different version of MCMC algorithm is necessary for Z. It turns out that the simple process that removes one current latent edge randomly and adds another one with the original acceptance-rejection algorithm works pretty well. However, the transition probabilities in both directions are not same :

$$P(\text{del } x, \text{add } y) = \frac{P_e(y)}{\sum_{y' \notin Z} P_e(y') + P_e(x)} \neq \frac{P_e(x)}{\sum_{y' \notin Z} P_e(y') + P_e(y)} = P(\text{del } y, \text{add } x)$$

Because of the asymmetric transition probability, the Metropolis-Hastings algorithm has to be introduced rather than simple Metropolis algorithm. Fortunately, most of terms are nicely canceled out; in the result, the accpetance rate is as follows :

$$\begin{aligned} &P(\text{Accept } Z' \to Z) \\ &= \min\left(1, \frac{P(x \in Z, y \notin Z) P(\text{del } x, \text{add } y)}{P(y \in Z', x \notin Z') P(\text{del } y, \text{add } x)}\right) \\ &= \min\left(1, \frac{P_e(x)(1-P_e(y)) \frac{P_e(y)}{\sum_{y' \notin Z} P_e(y') + P_e(y)}}{P_e(y)(1-P_e(x)) \frac{P_e(x)}{\sum_{y' \notin Z'} P_e(y') + P_e(x)}}\right) \\ &= \min\left(1, \frac{1-P_e(y)}{1-P_e(x)}\right) \end{aligned}$$

where $P_e(x)$ represents the probability that the edge x is connected.

In most cases, since P_e is very small, almost all transitions would be accepted. The entire modified E step is described in Algorithm A.6.

2.4 Maximization Step

If we try to calculate the gradient descent step with fixed samples of Z and σ, $O\left(S_{Gibbs}(N_0^k)\right.$ space is required to maintain them for Gibbs sample size S_{Gibbs}. Since the whole procedure is based on the Monte Carlo sampling, it would be reasonable to use the stochastic gradient descent step rather than deterministic one. In order to gain each gradient step of Θ, we resample Z and σ's with Θ_{old}, not with updated Θ. In this sense, each gradient update eventually repeats E step.

In the different point of view, the ECM algorithm[5] shows that only several steps of gradient descent algorithm would be enough. This technique is able to make the overall algorithm more efficient not only in time but also in space. The pesudocode is described in Algorithm A.7.

3. EXPERIMENTS

We have examined this fitting model for several kinds of dataset. With regard to the size of Kronecker initiator, Θ, it turns out that a 2×2 matrix would be fine[2]. Thus, in order to simplify the model, we used $N_0 = 2$ for every experiment. Where dataset is provided, we compare the KronEMFit to KronZeroFit.

Even though the both algorithms are scalable, some evaluation methods are not because, for example, it requires $O(N^2)$ time to compute the log-likelihood over only observed or latent graph. Therefore, for the purpose of practicable assessment, we simluate Kronecker graphs with thousands of nodes, or select a subgraph from the temporal dataset so that it should have several thousands of nodes.

Experiments consist of three major parts: we first generate Kronecker graph with a plausible initiator and measure many values on that graph and distribution in section 3.1. Then, the similar tests on real-world graphs are naturally followed in section 3.2. On the other hand, we are also interested in the evolution of Kronecker structure that the relate work is accomplished in section 3.3.

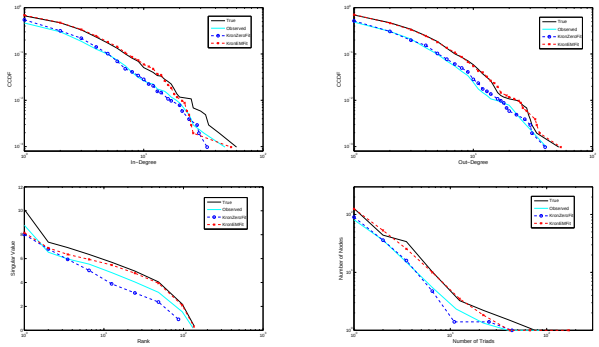

Figure 1: Network property plots for simulate Kronecker graph($N = 1,024$). KronEMFit(red-cross line) seems to have the same network properties as the true graph(black-solid line). Even though the number of nodes in KronZeroFit(blue-circle line) is 1,024, it looks closer to the observed graph(cyan-solid line) of size 768. The plot of top left, top right, bottom left, and bottom right indicates CCDF(Complementary CDF) of in-degree, CCDF of out-degree, singular value, and triad-participation, respectively.

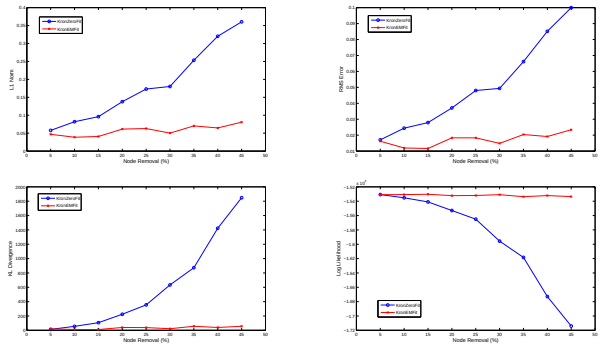

Figure 2: Each plot displays each metric (top left : absoulte error, top right : RMS error, bottom left : KL divergence, bottom right : log-likelihood) versus the percentage of node removal in simulate Kronecker graph of size 1,024. KronEMFit (red-cross line) shows steadiness regardless of how much the graph is removed, while KronZeroFit (blue-circle line) does not.

3.1 Synthesized Kronecker Graph

For the first step, to satisfy the latent setting, we generated a stochastic Kronecker graph($N = 1,024, E = 2,779, \Theta = [0.94\ 0.55;\ 0.57\ 0.15]$) and removed some nodes(25% of them) randomly. In this case, we are able to conjecture that the expected number of undeleted edges is approximately proportional to the square of the number of observed nodes. This assumption can be validated to see that the observed subgraph has 768 nodes and 1,649 edges. Table 1 describes the fitting results and various measurements over the graph. In any subgraph, with any metric, it shows that KronEMFit is closer to the true structure than KronZeroFit.

The outperformance can be also observed in various plots of network properties such as degree distribution. To draw these figures, we generated Kronecker graphs of size 1,024 with both estimated parameters and then produced features with them. In some sense, it might be natural since the absolute difference caused by KronEMFit is very small. Interestingly, KronZeroFit seems to capture the observed graph better, but we should note that the observed graph has 768 nodes but KronZeroFit contains 1,024. It eventually reveals the aspect of underestimation in KronZeroFit.

To verify better performance in general, we additionally carried out the same experiments by varying the percentage of deleted nodes. Surprisingly, parameters obtained by KronEMFit demonstrate its stability regardless of the number of removed nodes. Contrastively, KronZeroFit results in larger errors in proprotion to removal percentage as expected. Figure 2 displays the curves of errors, KL divergence, and log-likelihood against the true parameter.

3.2 Real-World Dataset

The same experiments were taken for real-world network. We chose the subgraph in the temporal dataset of LinkedIn so that the number of nodes should be 2,048. As we did in the previous section, we obtained the observed graph of $N = 1,536$ by erasing 25% of nodes randomly.

However, differently from the synthesized graph, it is impossible to know not only what is the true Kronecker parameter but also what is the true permutation. These limitations make the evaluation process difficult. Most measurements in table 1 would be invalid due to the lack of true initiator. In spite of its irrelevance to the true parameter, it is governed by the true permutation.

In this situation, the expected log-likelihood over the posterior distribution, $P(\sigma|T, \Theta)$, for the true graph T, might be an approximate measurment. This can be achieved by Monte-Carlo sampling, as done in the fitting process. Table 2 is obtained from KronZeroFit and KronEMFit in this method. Not surprisingly, KronEMFit results in smaller log-likelihood in any part of the graph than KronZeroFit, so we could assert that KronEMFit finds out closer Kronecker structure to the true one.

As plotted in section 3.1, several network features are also illustrated in figure 3. Since even more variation exists in the real network, it is impossible to track its properties strictly. However, one clear thing is that the curves of KronZeroFit is generally lower than that of KronEMFit and hold a little lighter tails. Seeing that the true real graph has the tendency to have heavier tails in many features, we can consider KronEMFit as a better model.

This assertion can be explained in a different point of view. When you look at the figure 3, there exists an additional (green) line unlike in the figure 1. It indecates TrueFit which is a Kronecker graph extracted from the true graph, not from the observed graph. It can be regarded as an approximation of the true parameter. Even in case that KronEMFit causes a little chasm between the real graph, it produces as almost same property-curves as TrueFit.

	ABS	RMS	KL	KL_O	KL_L	LL	LL_O	LL_L
KronZeroFit	0.173	0.048	354.4	206.2	148.2	-15,650	-9,182	-6,468
KronEMFit	0.063	0.018	22.4	13.3	9.1	-15,307	-8,989	-6,318
True	0.000	0.000	0.0	0.0	0.0	-15,295	-8,982	-6,313

Table 1: Results for simulated Kronecker graph of size 1,024. ABS and RMS indicate absolute errors and root-mean-square errors between the true and the estimated parameters, while KL, KL_O, and KL_L represent the Kullback-Leibler(KL) divergence between edge-distributions parameterized by estimated and true parameters on the overall, observed, and latent graph, repectively. Similary, LL, LL_O, and LL_L indicate the corresponding log-likelihoods. Every metric shows that KronEMFit outperforms KronZeroFit.

	LL	LL_O	LL_Z
KronZeroFit	-28389	-15735	-12645
KronEMFit	-27748	-15436	-12312

Table 2: Log-likelihood for LinkedIn graph of size 2,048

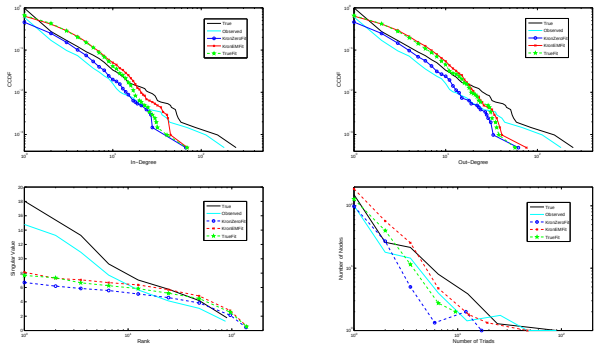

Figure 3: Network propertiy plots for LinedIn dataset. In every figure, KronZeroFit(blud-circle line) is likely to be lower as well as to drop earilier than KronEMFit(red-cross line). Also, it is remarkable that KronEMFit tends to follow the TrueFit which is the estimated Kronecker graph for the true graph. This illustrates that KronEMFit pulls out the Kronecker structure of the true graph. (Top left : In-degree CCDF, top right : out-degree CCDF, bottom left : singular value, bottom right : triad-participation)

Based on the assumption that TrueFit derives the Kronecker structure from the real graph, we can conclude that KronEMFit works well for the real dataset.

3.3 Graph Prediction

The tests mentioned above are handled in the snapshot of the given graph at the same time. However, we are able extend our approach to the evolution of the graph. The fact that Kronecker structure is consistent over time is already shown at the time when the number of nodes is almost equal to the power of N_0[2]. We hope that this consistency still holds even when the time is not equal to N_0^k.

We caught several snapshots of Yahoo-Flickr network between $N = 1,024$ and $N = 2,048$, made KronEMFit and KrnoZeroFit for $N_0 = 2$, and observed how they changed over time. However, for this experiment, a little different set-

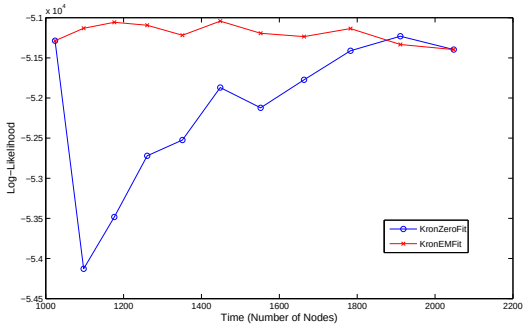

Figure 4: Log-likelihood for Flicker dataset of size 2,048. KronEMFit (red-cross line) looks stable in the log-likelihood for the future graph($N = 2,048$), while KronZeroFit (blue-circle line) does not. This tells that KronEMFit is more appropriate for the graph prediction than KronZeroFit.

ting is required. The assumtion about the expected number of latent edges would not be valid any longer in the temporal dataset; rather, the Densification Power Law(DPL)[3] can be a hint. Assuming that DPL is satisfies, the number of latent edges is expected to be :

$$\mathbf{E}[E_Z] = N_0^{k \log_N E} - E$$

for $k = \lceil \log_{N_0} N \rceil$.

Figure 4 represents the log-likelihood of each temporal Kronecker parameter for the future graph at $N = 2,048$. First, it is seen that Kronecker initiator at $N = 1,024$ works quite well for the graph at $N = 2,048$ as anticipated in [2]. Focusing on the time between $N = 1,024$ and $2,048$, the result from KronZeroFit suddenly falls down and increases gradually; on the other hand, KronEMFit tends to be stable in the log-likelihood sense. Moreover, their values are little different from that of the fitted Kronecker parameter at $N = 2,048$. From this point of view, we might claim that KronEMFit offers the comparatively persistent structure over time so that we could predict the graph in the future.

4. DISCUSSION

Section 3 confirms that KronEMFit leads a good inference of the Kronecker graph in the same snapshot as well as in the time series. By the way, someone might question why it is important to fit to the true Kronecker graph of size N_0^k, because the current graph contains only $N < N_0^k$ nodes. Moreover, KronZeroFit, the scheme of appending isolated nodes, sounds more attractive since it yields

the outcome much faster. Furthermore, it seems to capture the network properties of the observed graph, for instance, in the CCDF plots. In this sense, it could be said that KronEMFit chases the hypothetical structure.

However, since the Kronecker graph is mathematically defined only when $N = N_0^k$, it is indispensable to think of the Kronecker graph that covers the current one. The observed graph can be viewed as a sampled subgraph of the Kronecker graph. KronEMFit naturally gives us the inverse process of this sampling, which is the inference of the latent graph. Every intermidate stage is stochastically well-defined.

Therefore, despite of somewhat slowness and imaginariness, KronEMFit plays a key role on the Kronecker model. First of all, it is probablistically well-defined regardless of the existence of the closed-form. The main advantage is the consistency in some space, the Kronecker initiator. What we actually want in a stochastic graph model is the structure maintained globally in the given graph. This condition is satisfied in KronEMFit because the Kronecker parameter would not change a lot when we pick up a subgraph of the observed graph by deleting some nodes randomly. In contrast, zero-padding is not the reverse of any normal stochastic procedure.

Besides, the effect of underestimation in KronZeroFit arises due to the number of nodes, not due to the graph property. For an extreme example, if the current graph is the clique of size $N_0^{k-1}+1$, KronEMFit definitely outputs the complete structure, while KronZeroFit would not.

The inference becomes more substantial in the temporal dataset where it is equivalent to the prediction. Evidently, the addition of isolated nodes is not the solution for the graph predicton. It does not makes sense at all that the Kronecker parameter changes dramatically only because the number of nodes is just past the power of N_0. Further, it is general intuition that if the potential structure looks steady when $N = 1024, 2048, 4096$, and so on it should be also stable between those time ticks. KronEMFit furnishes this property, too.

Additionally, the probablistic model is good for us to apply lots variations on the model and develop it. For example, we could establish a new field in the graph theory by defining some function such as $Kron(G, N_0)$.

On the other hand, the Kronecker model originally includes some limitations. First, there is no reason to restrict the nodes to the permutation. The permutation indicates that any pair of nodes does not contain the exactly same combination of attributes, but it would not happen in the real world. We have not investigated what advantages we might gain by relaxing this limitation, but this must be a great topic.

In addition to the limitation, after taking some experiments with regard to link prediction, we figure out that high accuracy cannot be obtained by the current Kronecker model. It is probably because the model is parameterized by only a few values so that the edges could affect on each other too tightly. After all, the lack of flexibility can be the weakness of Kronecker model, and it is left for the future work.

5. CONCLUSION

In this paper, we introduced EM approach to estimate the potential Kronecker structure in any size of graph. Thanks to various MCMC techniques, the estimation can be tractable and scalabel. This fitting model provides not only a stochastically well-defined model but also the consistent parameters both in the randomly chosen subgraph and in the temporal graph.

6. REFERENCES

[1] W. K. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

[2] J. Leskovec, D. Chakrabarti, J. Kleinberg, C. Faloutsos, and Z. Ghahramani. Kronecker graphs: An approach to modeling networks. `arXiv:0812.4905v2 [stat.ML]`, 2008.

[3] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graph evolution: Densification and shirinking diameters. *ACM Transactions on Knowledge Discovery from Data*, 2007.

[4] J. S. Liu. Metropolized gibbs sampler: an improvement. Technical report, Dept. Statistics, Stanford University, 1996.

[5] X.-L. Meng and D. B. Rubin. Maximum likelihood estimation via the ecm algorithm: A general framework. *Biometrika*, 80(2):267–278, 1993.

[6] G. C. G. Wei and M. A. Tanner. A monte carlo implementation of the em algorithm and the poor man's data augmentation algorithms. *Journal of the American Statistical Association*, 85:699–704, 1990.

APPENDIX

A. ALGORITHMS

Algorithm A.1: BasicMCEM-EStep(G, Θ, T)

for $i := 1$ **to** T
  do $(Z^{(i)},\ \sigma^{(i)}) :=$ SamplePosterior(G, Θ)
$Z := (Z^{(1)}, \cdots, Z^{(T)})$
$\Sigma := (\sigma^{(1)}, \cdots, \sigma^{(T)})$
return (Z, Σ)

Algorithm A.2: BasicMCEM-MStep(G, Z, Σ, Θ)

repeat
 for $i := 1$ **to** T
   do $\Delta_i :=$ GradientDescentUpdate$(G, Z^{(i)}, \sigma^{(i)}, \Theta)$
 $\Delta := E(\Delta_i)$
 $\Theta := \Theta + \lambda\Delta$
until *converge*
return (Θ)

Algorithm A.3: SIMPLESAMPLEEDGE(G, σ, Θ, E_Z)

$Z_{ij} := 0$
repeat
  $(u, v) :=$ SELECTKRONECKEREDGE(σ, Θ)
  if $(u, v) \notin G$,
    then $Z_{uv} := 1$
until $\sum Z_{ij} = E_Z$
return (Z)

Algorithm A.4: SAMPLEEDGE$(G, \Theta, Z^{old}, \sigma)$

$Z := Z^{old}$
repeat
  $Z^{old} := Z$
  $x \sim$ uniformly selected in Z^{old}
  $Z^{old} := Z^{old} - \{x\}$
  $y \sim$ SIMPLESAMPLEEDGE$(G \cup Z^{old}, \Theta, \sigma)$
  $Z^{old} := Z^{old} \cup \{y\}$
  $u \sim U(0, 1)$
  if $u < \min\left(1, \frac{1-P_e(y)}{1-P_e(x)}\right)$
    then $Z := Z^{old}$
until several steps
return (Z)

Algorithm A.5: METROGIBBSSAMPLE$(G, \Theta, Z^{old}, \sigma^{old})$

$Z := Z^{old}, \sigma := \sigma^{old}$
$Z :=$ SAMPLEEDGE(G, Θ, Z, σ)
$\sigma :=$ SAMPLEPERMUTATION$((G \cup Z), \Theta, \sigma)$
return (Z, σ)

Algorithm A.6: NEWMCEM-ESTEP(G, Θ, T)

initialize $Z^{(0)}, \sigma^{(0)}$
for $i := 1$ **to** T
  do $(Z^{(i)}, \sigma^{(i)}) :=$ METROGIBBSSAMPLE$(G, \Theta, Z^{(i-1)}, \sigma^{(i-1)})$
$Z := (Z^{(1)}, \cdots, Z^{(T)})$
$\Sigma := (\sigma^{(1)}, \cdots, \sigma^{(T)})$
return (Z, Σ)

Algorithm A.7: NEWMCEM-MSTEP(G, Θ, Z, Σ)

$\Theta_{old} := \Theta$
repeat
  $(Z, \sigma) =$ NEWMCEM-ESTEP(G, Θ_{old}, T)**for** $i := 1$ **to** T
    do $\Delta_i :=$ GETGRADIENT$(G, Z^{(i)}, \sigma^{(i)}, \Theta)$
  $\Delta := E(\Delta_i)$
  $\Theta := \Theta + \lambda\Delta$
until *several steps*
return (Θ)