Graph Sparsification with Generative Adversarial Network

Hang-Yang Wu  
Department of Computer Science and Information Eng.  
National Taiwan University of Science and Technology  
Taipei, Taiwan  
M10615096@mail.ntust.edu.tw

Yi-Ling Chen  
Department of Computer Science and Information Eng.  
National Taiwan University of Science and Technology  
Taipei, Taiwan  
yiling@mail.ntust.edu.tw

Abstract—Graph sparsification aims to reduce the number of edges of a network while maintaining its accuracy for given tasks. In this study, we propose a novel method called GSGAN, which is able to sparsify networks for community detection tasks. GSGAN is able to capture those relationships that are not shown in the original graph but are relatively important, and creating artificial edges to reflect these relationships and further increase the effectiveness of the community detection task. We adopt GAN as the learning model and guide the generator to produce random walks that are able to capture the structure of a network. Specifically, during the training phase, in addition to judging the authenticity of the random walk, discriminator also considers the relationship between nodes at the same time. We design a reward function to guide the generator creating random walks that contain useful hidden relation information. These random walks are then combined to form a new social network that is efficient and effective for community detection. Experiments with real-world networks demonstrate that the proposed GSGAN is much more effective than the baselines, and GSGAN can be applied and helpful to various clustering algorithms of community detection.

Index Terms—Graph sparsification, Deep learning, Social network analysis

I. INTRODUCTION

There are numerous types of data that can be represented by graphs. For example, social networks, financial transaction networks, communication networks, and citation networks. As the size of data grows rapidly, it becomes impractical to analyze, store, and visualize such a large amount of network data. Thus, the technique of sparsification becomes more and more important. Graph sparsification can bring the following benefits: 1) After being sparsified, the graph size becomes smaller, and the storage space is thereby reduced. 2) Certain graph algorithms are not able to handle large graphs due to the high time complexity. However, if the graph is sparsified first, we can save considerable computation time and maintain the accuracy when applying these algorithms. 3) Graph sparsification can eliminate redundant data and noises from the graph. In addition to improving the performance of graph analysis algorithms, sparsification also makes the graph easier to be visualized. 4) When there are privacy concerns with the data, sparsification can remove and hide certain information from the graph to achieve better privacy protection.

The subsequent applications of graph sparsification can be diverse, leading to different requirements of sparsification, such as retaining specific structural patterns, paying attention to certain network entities, preserving graph query answers or maintaining the distribution of graph attributes. In general, the design of graph sparsification algorithms may involve the following four issues: 1) Different subsequent tasks favor different kinds of sparsified graphs. Fig. 1 is an illustrative example. For the influence maximization task, edge $a$ should be preserved in the sparsified graph; however, $a$ could be a redundant edge for the community detection task. 2) One of the main purposes of graph sparsification is to reduce the size of large input graphs. Therefore, the sparsification algorithms need to be capable of handling input graphs with large sizes. 3) In the real world, networks usually contain lots of noises that affect the performance of graph analysis. Therefore, it is also important for the sparsification algorithm to eliminate these noises, such that the performance of subsequent tasks can be further improved. 4) Previous studies of graph sparsification [1] [2] [3] usually focus on existing edges in the graph and make selections among them. However, this can lead to a key question: What if the edges not shown in the original graph contain more important information? For certain network analysis tasks, creating artificial edges for better results is allowed. If the perspective of generation can be incorporated (instead of only the perspective of elimination like the traditional methods), we will be able to generate new edges that do not exist in the original network but are essential for the subsequent tasks.

Example 1: Here we provide an example to better illustrate issue 4. Fig. 2(a) shows an example graph, where the blue nodes and yellow nodes represent different communities. If we replace some existing edges between different communities (e.g., edges $a$ and $b$) with some artificial edges inside the community (e.g., the dotted lines in Fig. 2(b)), the community structure would become more compact under the same number of edges, which is conducive to the community detection task. In the traditional sparsification methods, this kind of artificial edges are not considered; by contrast, the proposed method in this study is
able to create artificial edges that do not exist in the original network but contain important information for the given task during the graph sparsification.

In this study, we propose a novel graph sparsification method called Graph Sparsification with Generative Adversarial Network (GSGAN) to address the above four issues. For the issue 1, GSGAN can flexibly replace components (i.e., reward functions) to provide different directions of sparsification that fit the subsequent tasks. For the issue 2, GSGAN divides the network into multiple random walks, in order to be capable of processing huge data. For the issue 3, GSGAN can accurately filter out noises and focus on retaining important information in the sparsified graph. For the issue 4, GSGAN is able to capture those relationships that are not shown in the original network but are relatively important, and creating artificial edges to further increase the effectiveness of the community detection task.

We use GAN [4] as the learning model and guide Generator $G$ to produce random walks that capture the structure of the network. Specifically, during the training process, in addition to judging the authenticity of the random walk, Discriminator $D$ also considers the relationship between nodes at the same time.

We design a reward function to guide the generator creating random walks that contain useful hidden relation information. These random walks are then combined to form a new social network that is efficient and effective for community detection. Experiments with real-world networks demonstrate that the proposed GSGAN is much more effective than the baselines. Moreover, the execution time of the network generated by GSGAN is still capable of reaching acceptable community detection results.

We observe that previous studies only focus on the existing edges and make selections from them, but for certain scenarios, such as community detection, it is allowed to create additional edges for better results. Therefore, our GSGAN is designed to be able to create artificial edges and further increase the effectiveness of community detection tasks.

We compare the proposed GSGAN with other baselines with various network datasets and different clustering algorithms of community detection. The experimental results show that the sparsified graphs generated by GSGAN lead to community detection results significantly superior to the sparsified graphs obtained from the baseline sparsification algorithms. It is worth noting that GSGAN is able to handle a much wider range of ratios than other baseline methods. When the ratio is low (e.g., 1%), only the sparsified network generated by GSGAN is still capable of reaching acceptable community detection results.

Despite of only using far fewer edges (e.g., 5%), applying clustering algorithms of community detection on our sparsified graph receives comparable or even better results than on the original graph. Moreover, the execution time of community detection can be considerably reduced (e.g., by nearly an order of magnitude) when applying on the sparsified graph.

The rest of the paper is organized as follows. Section 2 surveys the related works. Section 3 defines the problem and introduces the proposed GSGAN. Section 4 shows the experimental setup and results. Finally, we conclude our study in Section 5.

II. RELATED WORKS

In this section, we introduce and discuss some previous works related to graph sparsification. Moreover, we also introduce the studies related to network generation, since our sparsification algorithm also generates some important artificial edges.

A. Graph Sparsification

Sparsification has been extensively studied and applied to different domains, such as community detection, influence maximization and retaining certain network properties. Sparsification can be roughly divided into two categories based on their goals: (1) preserving general network properties and (2) application-based sparsifiers. Our proposed GSGAN is more application-based. Therefore, we focus on introducing application-based sparsifiers here and provide details about preserving general network properties in [5].

Application-based sparsifiers is to do sparsification for specific tasks, such as community detection, influence maximization, link prediction. It is worth noting that different tasks require different graph sparsification. For link prediction, DEDS [6] processes the original graph into multiple smaller networks to improve the efficiency of link prediction. SPINE [7] mainly do sparsification for influence maximization. It uses a maximum likelihood method to find a set of edges that generates an observed set of influence cascades with the greatest probability. L-Spar [1] develops a sparsification algorithm specifically for graph clustering. The underlying principle is to preferentially retain edges between nodes that are in the same cluster by selecting edges between nodes with similar neighbors. LD [2] proposes some heuristic methods aiming to preserve the structure of social networks. Their main method Local Degree keeps only the edges to hubs, i.e., vertices with high degree, claiming that they are crucial for the topology of complex social networks. NETSPARSE [3] proposes a new formulation of network sparsification, where the input consists not only of a network but also of a set of communities.

B. Network Generation

Typically, a social network is represented by the adjacency matrix. In [8], the authors apply GAN [9] to graph data by
trying to directly generate an entire matrix. However, it is not suitable to use deep learning to directly generate an entire matrix, which is limited by the size of the matrix. To address this limitation, NETGAN [10] tends to capture the topology of the social network by random walks. Intuitively, NETGAN already has the effect of preserving important edges, since the edges of the random walks are sampled according to their frequency when the random walks are combined to form a whole social network. Theoretically, more important edges will appear more often in random walks. Therefore, using NETGAN architecture directly may also produce sparsified graph \( G' \), but there are no guarantees that this \( G' \) will be effective for the analysis of community detection. Inspired by NETGAN, we use random walks to do graph sparsification.

III. APPROACH

In Section 3.1, we first define the problem. Section 3.2 then introduces our GSGAN model, and Section 3.3 explains the training process. In Section 3.4, we introduce the proposed measurements, and Section 3.5 details how to use GSGAN model to form the sparsified network.

A. Problem Definition

Given a static, undirected graph \( G(V,E) \), where \( V \) represents the set of nodes in the graph and \( E \) represents the set of edges in the graph. The objective of GSGAN is to find a sparsified graph \( G'(V,E') \) with fewer edges, while the quality of community detection results is retained or even improved.

Please note that in the sparsified graph \( G' \), we may add some new and important artificial edges. One important contribution of this study is that we address the issue illustrated in Example 1 by introducing the artificial edges. As show in the experiments, these artificial edges are quite helpful for community detection. To the best of our knowledge, this artificial edge mechanism has not been considered in previous studies of graph sparsification.

B. Model

In this subsection, we provide the details of the proposed method, Graph Sparsification with Generative Adversarial Network (GSGAN). GSGAN consists of three components: 1) GAN, 2) LSTM and 3) Policy Gradient. We use GAN [9] as the learning model to learn random walks that conform to the network structure of the original network. As the random walk contains an ordered sequence of nodes, both the generator and the discriminator in GAN use LSTM [11] structure with time series memory. In addition, we also apply policy gradient to guide our model towards a specific direction through reward, then performing graph sparsification for community detection. In the following, we introduce the proposed GSGAN and its components in a bottom-up way.

GAN: The general idea of GAN is to simultaneously train a generator \( G \) and a discriminator \( D \) with conflict objectives. The generator \( G \) tries to capture data distribution of the training set, and the discriminator \( D \) tries to identify samples produced by the generator from real ones. That is, \( G \) and \( D \) play the following two-player min-max game:

\[
\min_G \max_D V(D,G) = \mathbb{E}_{x \sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))]
\]  

LSTM: LSTM is controlled by input and output gates which determine whether to update the hidden layer state. While RNN [12] only has a single transmission state, LSTM has two cell states \( c_t \) and \( h_t \), where \( c_t \) changes slowly and \( h_t \) changes drastically. The output \( y_t \) is determined by \( c_t^{-1}, h_t^{-1} \) and the current input.

\[
c_t = z_t \odot c_t^{-1} + z_t \odot z
\]

\[
h_t = z_t \odot \text{tanh}(c_t)
\]

\[
y_t = \sigma(W h_t)
\]

Policy gradient: Reinforcement learning guides the learning goals through the rewards obtained from the interaction between action and environment. Policy gradient (PG) [13] is a method to achieve reinforcement learning. The core concept of PG is to maximize the rewards obtained by performing action \( a_t = \{a_1, a_2, ..., a_n\} \) in the environment \( s_t = \{s_1, s_2, ..., s_n\} \). Before updating the network parameters, the reward of the sample will be calculated in advance. When updating the network parameters, the reward is considered as the weight of the loss.

\[
\nabla R_{\theta} = \sum_\tau R(\tau) \nabla p_{\theta}(\tau) = \mathbb{E}_{(a_t|s_t) \sim p_{\theta}(a_t|s_t)}[R(a_t|s_t) \nabla \log p_{\theta}(a_t|s_t)]
\]

GSGAN: An overview of GSGAN architecture and the internal structure of generator are provided in Fig. 3. Here we use WGAN [14] instead of the traditional GAN architecture, because WGAN is more effective during the learning process. The interior of the generator is mainly the LSTM structure. The number of LSTM layers is the same as the length of the random walks to be generated. For the generation process, a latent vector \( z \) drawn from a normal distribution is passed through a dense matrix \( W \in \mathbb{R} \) to get initial values \((c^{0}, h^{0})\).

At each time step \( t \), to generate the next node in the random walk, the output of LSTM cell will refer to \( c^{t-1}, h^{t-1} \) and the previous node, and then map to a probability vector \( p_t = \{p_1, p_2, ..., p_n\} \) through a fully connected dense \( W \), where \( W \in \mathbb{R}^{d \times N} \). We select the node corresponding to the maximum value in \( p_t \) as the current node \( v_t \). The above process repeats until the random walk reaches the required length. After the random walk is generated, we update network parameters with the policy gradient. The social network graph is considered as the environment, and the generation of edges in the random walk is considered as actions. We score the importance for each edge in the random walk and sum up these scores as a reward for generating this random walk. Let \( RW, T, SN \) and \( e \) denote the random walk, the length of random walk, the social network and the edge of random walk, respectively, then the reward can be formulated as:

\[
f_{\text{reward}}(RW) = \begin{cases} 
\sum_{i=1}^{T} f_{\text{score}}(SN,e_i) & \text{if } L_G < 0 \\
1 & \text{else}
\end{cases}
\]
More details about the reward function will be provided later in Section 3.4.

The loss of each random walk will be multiplied by the corresponding reward to guide the network update such that the model tends to generate high-importance edges. We leverage this approach to generate edges that benefit the community detection task, even though the edge might not appear in the original network. The loss function of generator \( G \) and discriminator \( D \) are as follows:

\[
L_G = -\frac{1}{m} \sum_{i=1}^{m} D(G(z^{(i)})) \ast f_{\text{reward}}(G(z^{(i)})) \tag{7}
\]

\[
L_D = \frac{1}{m} \sum_{i=1}^{m} D(x^{(i)}) - \frac{1}{m} \sum_{i=1}^{m} D(G(z^{(i)})) \tag{8}
\]

In the next subsection, we will introduce how to train the GSGAN model.

C. Training

We first sample a batch of noises from normal distribution as the generator’s input. After the generator outputs random walks, we calculate the score of each edge on this random walk. The goal here is to guide the generator to produce certain edges even when these edges receive low ratings from the discriminator (indicating that these edges do not exist in the original network), since these artificial edges are highly beneficial for community detection tasks. We sum up the score of each edge as a reward for this random walk, and multiply the loss obtained from each random walk by the corresponding reward. Even the discriminator gives a low rating to this random walk, the value may be raised after multiplying the reward. The advantage of this mechanism is that GSGAN can generate the edges important to the subsequent tasks (e.g., community detection), even though the edge may not exist in the original network. Due to space constraints, please see [5] for the pseudocode. Next, we will introduce a measurement related to community detection and its improved version as our reward function.

D. Reward

Most methods for graph sparsification focus on retaining important edges among the existing ones, such as [15] [1] [2]. There are many ways to measure the importance of edges. For example, the Jaccard coefficient, edge betweenness centrality and clustering coefficient, but not all measurements are helpful for the community detection task. From the aspect of community detection, keeping intra edges in the community should have higher priority over inter edges between communities. A commonly accepted assumption is that if there are more common neighbors between two nodes, they are more likely to be connected by a link, and hence they also have a higher chance to be in the same community. Following this idea, we first introduce the Jaccard coefficient measurement and then propose a modified version for better evaluating the importance of an edge to the community detection task.

**Jaccard coefficient:** Jaccard coefficient is a popular measurement to compare the similarity and diversity of a sample set. The similarity between two nodes can be measured using Jaccard coefficient as Equation 9, where \( N_u \) and \( N_v \) denote the neighborhood of node \( u \) and \( v \), respectively

\[
Jaccard(u, v) = \frac{|N_u \cap N_v|}{|N_u \cup N_v|} \tag{9}
\]

**Density Jaccard:** One problem of the aforementioned basic Jaccard coefficient is lacking the consideration of cluster density. In Fig. 2(c), there are two communities (i.e., the eight nodes on the top, and the four nodes in the bottom). When calculating the Jaccard coefficient of an edge \((u, v)\), the edge in the community above tends to be considered as more important than the edge in the community below. Therefore, using Jaccard coefficient to measure the importance of edges would easily cause the community below losing most of its edges and becoming a bunch of isolated points. To address this problem, we improve the basic Jaccard coefficient to take the density into consideration. Let \( G_N(u, v) \) be the subgraph formed by the neighbors of \( u \) and \( v \), its density is defined as \( D(G_N(u, v)) = \frac{2|E_N(u, v)|}{|V_N(u, v)||V_N(u, v)| - 1} \), where \(|V_N(u, v)|\) and \(|E_N(u, v)|\) is the number of nodes and the number of edges in the subgraph \( G_N(u, v) \), respectively. The Density Jaccard coefficient of an edge \((u, v)\) is then defined as follows:

\[
\text{DensityJaccard}(u, v) = \frac{|N_u \cap N_v|}{|N_u \cup N_v| \ast D(G_N(u, v))} \tag{10}
\]

In GSGAN, the above measurements are used in the reward function to calculate the importance of edges. With these measurements, when the discriminator \( D \) evaluates each edge
in the random walk, it not only considers the correctness but also the relationship between the nodes at the two ends of that edge, and thus capturing the important relationships represented by the artificial edges.

E. Assembling Social Network

Since the final output of GSGAN should be a sparsified network rather than some random walks, we still need to transform the generated random walks into a social network. First, the trained generator $G$ produces a large number of random walks. After counting the number of times each edge appearing in these random walks, we can obtain a matrix $S$, where each entry in $S$ represents the number of times that an edge appears. After divided by the total number of edges, $S$ becomes a probability matrix. Next, GSGAN starts to choose suitable edges based on $S$. A simple method is to set a threshold or directly select top-k edges, but this kind of approach may lead to the lost of low-degree nodes or even produce many singletons. Other studies also emphasize the importance of network connectivity [16] [17]. Therefore, we adopt a more sophisticated selection strategy. We first pick at least one edge for each node. Then, if the number of edges in the generated target graph $G_T$ is fewer than the desired number of edges $d$, we continue sampling the edge $(i, j)$ with probability $p_{ij} = \frac{S_{ij}}{\sum_{l \in \text{out}(i)} S_{il}}$. Otherwise, if the number of edges already exceeds $d$, we will sequentially remove the edges according to the probability $p_{ij}$ (from low to high). Please see [5] for the pseudocode of the above procedure.

IV. EXPERIMENTS

In the experiments, we evaluate and compare the proposed GSGAN with six baseline methods with three real-world datasets under three classic clustering algorithms of community detection, including Clauset [18], label propagation [19], and Louvain [20]. We also perform cross-comparisons of with and without artificial edges. Finally, we visualize the sparsified graph. Due to space constraints, we only present part of results here. Please refer to the full version [5] for complete experimental results and the description of baseline methods and datasets.

A. Comparison with baselines

We present the experimental results in curve graphs, where different methods are distinguished by different colors. Note that the grey dotted line represents the standard line, which is the accuracy of community detection using the original network without sparsification. In Fig. 4(a), it can be seen that the proposed GSGAN defeats all the baselines. At ratio 30 (i.e., the number of edges in the sparsified network is only 30% of that number in the original network), GSGAN even exceeds the standard line, which means that the network after sparsification has better community detection performance than the original network. As mentioned earlier, the real-world network tends to have noises, and the sparsification can help reduce these noises and bring better performance. At ratio 10, our method still maintains performance similar to the standard line, while other baselines have lost their usefulness.

The performance of our proposed GSGAN in Fig. 4(b) is higher than the standard line for almost all ratios. Although L-spar reaches a peak and exceeds GSGAN around ratio 10 in this dataset, overall the proposed GSGAN still has clear advantages for most of the ratios and datasets. Note that even at ratio 1, our method still maintains competitive performance compared to the original network, while other baselines almost cannot work at such a low ratio. Due to that L-spar, LD and NETPARSE are not designed to follow any given network size, we adjust the parameters of these methods such that the numbers of edges in the sparsified graphs generated by these methods are consistent with the number of edges in our sparsified graph. Note that some methods cannot cover ratio 1, so we only present the results up to their limits.

We also measure the time reduction of community detection algorithms using the sparsified graphs generated by GSGAN. The execution time of Louvain at ratio 10 of the Youtube social network, ratio 10 of Eumail network and ratio 10 of Facebook network is reduced by around 85%, 50% and 65%, respectively. More detailed analysis on time reduction can be found in [5].

B. Analysis of artificial edges

In order to evaluate how helpful the artificial edges are to the community detection task, we conduct experiments using the sparsified networks with artificial edges and without artificial edges. Specifically, to generate a sparsified network without the artificial edges, we skip artificial edges and only choose edges that already exist in the network during the edge selection process. In Fig. 5, it can be seen that if the artificial edges are not adopted, the performance tends to decline at all the ratios. Such performance drop appears in every dataset, especially in the Youtube social network (i.e., Fig. 5(a)).

C. Graph visualization

Comparing Fig. 6(a) and Fig. 6(b), it can be found that GSGAN tends to eliminate the edges between different communities, and the artificial edges are used to make the structure inside the community more compact. Although JC still roughly
low ratios.

various community detection algorithms, especially at extremely

show that our method is quite effective and efficient under

algorithms of community detection. The experimental results

in order to further improve the performance of community

able to create and add artificial edges to the sparsified network

but still contain important information. That is, GSGAN is

also considers the edges that do not exist in the original network

serving essential edges from the original network, our method

important for the community detection task. In addition to pre-

to capture the structure of networks and retain edges that are

rithms called GSGAN. Our proposed method uses random walks

separates the two communities, the community detection using

sparsified networks generated by JC does not perform well. The

red nodes in Fig. 6(c) represent the nodes that are not in the

community. On the contrary, the community detection with the sparsified graphs generated by GSGAN is able to locate

most of the nodes in the correct community. It should be noted

that the number of edges in Fig. 6(b) seems to be larger than

Fig. 6(c), but in the entire sparsified graph, the total number

of edges selected by GSGAN and JC are the same. That is, GSGAN is able to spend most of the quota on the edges inside the

community rather than the edges between the communities.

V. CONCLUSIONS

In this study, we propose a novel graph sparsification algo-

rithms called GSGAN. Our proposed method uses random walks
to capture the structure of networks and retain edges that are

important for the community detection task. In addition to pre-

serving essential edges from the original network, our method
also considers the edges that do not exist in the original network
but still contain important information. That is, GSGAN is

able to create and add artificial edges to the sparsified network
in order to further improve the performance of community
detection. We evaluate the proposed method and other baselines
using three real-world social networks under three clustering
algorithms of community detection. The experimental results
show that our method is quite effective and efficient under various community detection algorithms, especially at extremely

low ratios.

REFERENCES

[1] V. Satuluri, S. Parthasarathy, and Y. Ruan, “Local graph sparsification

for scalable clustering,” in Proceedings of the 2011 ACM SIGMOD

International Conference on Management of data. ACM, 2011, pp. 721–
732.

[2] M. Hamann, G. Lindner, H. Meyerhenke, C. L. Staudt, and D. Wagner,

“Structure-preserving sparsification methods for social networks,” Social

Network Analysis and Mining, vol. 6, no. 1, p. 22, 2016.

[3] A. Gionis, P. Rozenstein, N. Tatti, and E. Terzi, “Community-aware

network sparsification,” in Proceedings of the 2017 SIAM International

Conference on Data Mining. SIAM, 2017, pp. 426–434.

[4] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S.

Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” in

Advances in neural information processing systems, 2014, pp. 2672–2680.

[5] H.-Y. Wu and Y.-L. Chen, “Graph sparsification with generative adversarial

network,” arXiv preprint arXiv:2009.11736, 2020.

[6] Y.-L. Chen, M.-S. Chen, and S. Y. Philip, “Ensemble of diverse spar-

sifications for link prediction in large-scale networks,” in 2015 IEEE

International Conference on Data Mining. IEEE, 2015, pp. 51–60.

[7] M. Mathioudakis, F. Bonchi, C. Castillo, A. Gionis, and A. Ukkonen,

“Sparsification of influence networks,” in Proceedings of the 17th ACM

SIGKDD international conference on Knowledge discovery and data

mining. ACM, 2011, pp. 529–537.

[8] S. Tavakoli, A. Hajibagheri, and G. Sukthankar, “Learning social graph
topologies using generative adversarial neural networks,” in International

Conference on Social Computing, Behavioral-Cultural Modeling & Pre-
diction, 2017.

[9] A. Radford, L. Metz, and S. Chintala, “Unsupervised representation

learning with deep convolutional generative adversarial networks,” arXiv
preprint arXiv:1511.06434, 2015.

[10] A. Bogchenski, O. Shchur, D. Zitniger, and S. Günneweim, “Netgan:

Generating graphs via random walks,” arXiv preprint arXiv:1803.00816,
2018.

[11] M. Sundermeyer, R. Schlüter, and H. Ney, “Lstm neural networks for

language modeling,” in Thirteenth annual conference of the international

speech communication association, 2012.

[12] T. Mikolov, M. Karafiát, L. Burget, J. Černocký, and S. Khudanpur,

“Recurrent neural network based language model,” in Eleventh annual

conference of the international speech communication association, 2010.

[13] P. Dayan and B. W. Balleine, “Reward, motivation, and reinforcement

learning,” Neuron, vol. 36, no. 2, pp. 285–298, 2002.

[14] M. Arjovsky, S. Chintala, and L. Bottou, “Wasserstein gan,” arXiv preprint

arXiv:1701.07875, 2017.

[15] P. Wong, C. Sun, E. Lo, M. L. Yu, X. Wu, Z. Zhao, T.-H. H. Chan, and

B. Kao, “Finding k most influential edges on flow graphs,” Information

Systems, vol. 65, pp. 93–105, 2017.

[16] F. Zhou, S. Malher, and H. Toivonen, “Network simplification with

minimal loss of connectivity,” in 2010 IEEE international conference on
data mining. IEEE, 2010, pp. 659–668.

[17] F. Zhou, S. Malher, and H. Toivonen, “Simplification of networks by edge

pruning,” in Bisociative Knowledge Discovery. Springer, 2012, pp. 179–
198.

[18] A. Clauset, M. E. Newman, and C. Moore, “Finding community structure in

very large networks,” Physical review E, vol. 70, no. 6, p. 066111, 2004.

[19] G. Cordasco and L. Gargano, “Community detection via semi-synchronous

label propagation algorithms,” in 2010 IEEE International Workshop on:

Business Applications of Social Network Analysis (BASNA). IEEE, 2010,
pp. 1–8.

[20] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, “Fast

unfolding of communities in large networks,” Journal of statistical me-

chanics: theory and experiment, vol. 2008, no. 10, p. P10008, 2008.