Appendix of *GraphSAID*: *Graph S*ampling via *A*ttention based *I*nteger Programming Metho*d*

Ziqi Liu
New York University Center for Data Science
New York, United States
ziqiliu@nyu.edu

Laurence Liu
FCC Analytics
Hong Kong, China
laurence0636@outlook.com

1 Full Algorithmic Description of *Attended Connected Component Generation*.

Algorithm 1 Attended Connected Component Generation

Require: $\mathcal{G}, \tau, E$
Ensure: $\mathcal{L}$ (the collection of connected components as node sets)

List $\mathcal{L} \leftarrow [\{\}]$
Queue $Q \leftarrow E$
while $Q \neq null$ **do**
  Edge $e \leftarrow Q.pop()$
  $s_1 \leftarrow (s \in \mathcal{L}\ s.t.\ e[0] \in s)$
  $s_2 \leftarrow (s \in \mathcal{L}\ s.t.\ e[1] \in s)$ ▷ $e[0]$, $e[1]$ are vertexes
  if $s_1 = null$ and $s_2 = null$ **then**
    $s^{new} \leftarrow \{e[0], e[1]\}$
    $\mathcal{L}.append(s^{new})$
  else if $s_1 \neq null$ or $s_2 \neq null$ **then**
    $s_{found} \leftarrow (s \in \{s_1, s_2\}\ s.t.\ s \neq null)$
    $n_{found} \leftarrow (n \in \{e[0], e[1]\}\ s.t.\ n \in s_{found})$
    $n_{new} \leftarrow (n \in \{e[0], e[1]\}\ s.t.\ n \notin s_{found})$
    if $|s_{found}| < \tau$ **then**
      $s_{found} = s_{found} \bigcup \{n_{new}\}$
      update $\mathcal{L}$ with s^{found}
    else
      $s^{new} \leftarrow \{n_{new}\}$; $\mathcal{L}.append(s^{new})$
    end if
  else if $s_1 \neq null$ and $s_2 \neq null$ **then**
    $s_1 \leftarrow (s \in \mathcal{L}\ s.t.\ e[0] \in s)$
    $s_2 \leftarrow (s \in \mathcal{L}\ s.t.\ e[1] \in s)$
    if $s_1 \neq s_2$ **then**
      $s^{merge} \leftarrow s_1 \bigcup s_2$
      remove s_1, s_2 in $\mathcal{L}$; $\mathcal{L}.append(s^{merge})$
    end if
  end if
end while

2 Full Description of Datasets

GRPG. There are many graph generators can model a transaction network, e.g., [6, 3, 4, 1, 2, 5]. In this work, we model our synthesized network as a realization of a *Gaussian Random Partition Graph* process [2], due to the distinct nature of intra- and inter-group activities which can be used to model complex networks like cash flow. The *Gaussian Random Partition Graph* (GRPG) process can be noted as $G(n, s, v, p_{in}, p_{out})$, where n is the number of nodes, s is the mean of the cluster size, s/v is the variance of the cluster size, p_{in} and p_{out} are the probabilities of intra- and inter-cluster connection. Figure 1 shows visualization of graph structure changes by controlling different p_{out}. In this example,

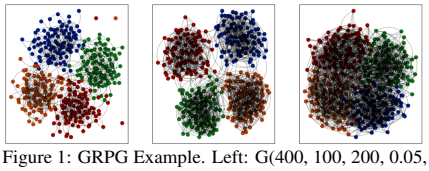

Figure 1: GRPG Example. Left: G(400, 100, 200, 0.05, 0.001); Middle: G(400, 100, 200, 0.1, 0.001); right: G(400, 100, 200, 0.1, 0.01).

		Elliptic			Cora		Citeseer		GPRG	
B	Method	R_{attn}	Acc	F1	R_{attn}	Acc	R_{attn}	Acc	R_{attn}	Acc
60%	FF	0.156	0.940	0.474	0.248	0.747	0.628	0.618	0.434	0.832
60%	SB	0.157	0.934	0.481	0.253	0.788	0.645	0.589	0.402	0.856
60%	SRW	0.158	0.941	0.517	0.250	0.806	0.645	0.578	0.407	0.861
60%	RWF	0.146	0.949	0.544	0.249	0.811	0.647	0.594	0.409	0.861
60%	ORC-sub	oom	oom	oom	0.258	0.743	0.646	0.618	0.483	**0.862**
60%	Ours	**0.506**	0.952	0.557	**0.380**	0.815	**0.650**	**0.626**	**0.627**	0.860
60%	*Ours*†	**0.506**	0.951	0.557	**0.380**	**0.817**	**0.650**	0.625	**0.627**	0.860
60%	*Ours*‡	**0.506**	**0.952**	**0.559**	**0.380**	0.815	**0.650**	**0.626**	**0.627**	0.860
80%	FF	0.198	0.950	0.537	0.344	0.831	0.857	0.626	0.660	0.849
80%	SB	0.213	0.929	0.412	0.351	0.828	0.861	0.627	0.674	0.860
80%	SRW	0.193	0.948	0.524	0.348	0.834	0.873	0.635	0.672	**0.861**
80%	RWF	0.202	**0.955**	**0.557**	0.351	0.832	**0.870**	0.629	0.669	**0.861**
80%	ORC-sub	oom	oom	oom	0.353	0.811	**0.870**	0.627	0.703	**0.861**
80%	Ours	**0.617**	**0.955**	0.540	**0.495**	0.837	**0.870**	0.629	**0.822**	**0.861**
80%	*Ours*†	**0.617**	**0.955**	0.545	**0.495**	**0.838**	**0.870**	0.634	**0.822**	**0.861**
80%	*Ours*‡	**0.617**	**0.955**	0.540	**0.495**	0.836	**0.870**	**0.636**	**0.822**	**0.861**

Table 1: Learning performance comparison using different sampled subgraphs. The best performance is in **bold**. "oom" means out-of-memory issue during the computation.

four partitions are labeled in colors, while from left to right the difficulty to detect the partitions by graph structure increases.

Citeseer. The CiteSeer dataset consists of 3312 scientific publications classified into one of six classes. The citation network consists of 4732 links. Each publication in the dataset is described by a 0/1-valued word vector indicating the absence/presence of the corresponding word from the dictionary. The dictionary consists of 3703 unique words.

Cora. The Cora dataset consists of 2708 scientific publications classified into one of seven classes. The citation network consists of 5429 links. Each publication in the dataset is described by a 0/1-valued word vector indicating the absence/presence of the corresponding word from the dictionary. The dictionary consists of 1433 unique words.

Elliptic. The Elliptic dataset maps Bitcoin transactions to real entities belonging to licit categories (exchanges, wallet providers, miners, licit services, etc.) versus illicit ones (scams, malware, terrorist organizations, ransomware, Ponzi schemes, etc.). It consists of 203,769 nodes and 234,355 transactions as edges. In addition, 157,205 nodes are unlabelled.

2.1 Full Description of GRPG dataset synthesis

We synthesize our GRPG dataset by setting: *(1)* $G(1000, 200, 100, 0.05, 0.005)$; and *(2)* let the 5 expected clusters have a $\mathcal{N}_{k=3}(\mu_i, \Sigma_i)$, where $i \in \{1, 2, 3, 4, 5\}$, $\mu_i = (i-3) \cdot \mathbf{1}_k^T$ is the mean vector for the *ith* partition and $\Sigma_i = diag(\mathbf{1}_k^T)$ for all partitions. We denote node feature for node j as v_j, and $p_j \in \{1, 2, 3, 4, 5\}$ is the the partition for node j. To make our toy dataset more realistic, we set 25% of nodes as "unknown" (i.e., "-1"). Then we define the node labels of the other 75% nodes as:

$$label_j = \begin{cases} 0, & \text{if } \mathbf{1}^T v_j \leq p_j \\ 1, & \text{otherwise} \end{cases} \tag{1}$$

Table 2 summarizes the label distribution in our GRPG dataset.

block	1			2			3			4			5		
label	-1	0	1	-1	0	1	-1	0	1	-1	0	1	-1	0	1
count	43	156	2	63	119	20	46	74	80	58	15	125	40	2	156

Table 2: Statistics of toy datast *GRPG*.

References

[1] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *science*, 1999.

[2] Ulrik Brandes, Marco Gaertler, and Dorothea Wagner. Experiments on graph clustering algorithms. In *European symposium on algorithms*, 2003.

[3] Sergey N Dorogovtsev, Alexander V Goltsev, and José Ferreira F Mendes. Pseudofractal scale-free web. *Physical review E*, 2002.

[4] Paul Erdos, Alfréd Rényi, et al. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci*, 1960.

[5] Paul W Holland, Kathryn Blackmond Laskey, and Samuel Leinhardt. Stochastic blockmodels: First steps. *Social networks*, 1983.

[6] Jaewon Yang and Jure Leskovec. Community-affiliation graph model for overlapping network community detection. In *Data Mining*, 2012.