Distributed Generation of Billion-node Social Graphs with Overlapping Community Structure

Kyrylo Chykhradze*, Anton Korshunov*, Nazar Buzun*, Roman Pastukhov*, Nikolay Kuzyurin*, Denis Turdakov*, and Hangkyu Kim**

*Institute for System Programming of the Russian Academy of Sciences
Moscow, Russia
**Data Intellegence Lab, Software Research Center, Samsung Electronics Co., Ltd.
Suwon, South Korea
{chykhradze, korshunov, nazar, pastukhov, nnkuz, turdakov}@ispras.ru
hangkyu.kim@samsung.com

Abstract. In the field of social community detection, it is commonly accepted to utilize graphs with reference community structure for accuracy evaluation. The method for generating large random social graphs with realistic community structure is introduced in the paper. The resulting graphs have several of recently discovered properties of social community structure which run counter to conventional wisdom: dense community overlaps, superlinear growth of number of edges inside a community with its size, and power law distribution of user-community memberships. Further, the method is by-design distributable and showed near-linear scalability in Amazon EC2 cloud using Apache Spark implementation.

Keywords: random graph, social network, community detection, benchmark network, graph generation, LFR benchmark, Affiliation Graph Model, SNAP, distributed algorithms, Amazon EC2, Apache Spark

1 Introduction

Community structure is a natural property of human networks, including online social networks where users tend to unite either explicitly (by means of grouping functionality of network software) or implicitly (by establishing ties based on shared affiliation, role, activity, social circle, interest, function, or some other property). Social data scientists widely employ intuitive notions of separability, density, and cohesiveness of social groups for discovering and evaluating implicit communities from social networks [4,10,12].

Recent advances in studying modular structure of social networks [13,14] helped to reveal several fundamental properties that appear to be common in human interaction networks: dense community overlaps, superlinear growth of number of edges inside a community with its size, power law distribution of user-community memberships and communities size, etc. This suggests the need for revisiting accuracy evaluation techniques for community detection methods and adequacy of the methods themselves.

Despite the availability of community detection benchmarks based on real networks, it is desirable to learn some fundamental properties from it and develop

a tool for producing synthetic benchmarks with similar properties and different characteristics. For a reliable and comprehensive evaluation, a community detection method must be tested on benchmark networks of variable size and other parameters as they may have significant impact on the results.

The main contributions of the paper could be summarized as follows:

- we introduce a novel approach to benchmark networks generation for community detection methods based on *Community-Affiliation Graph Model (AGM)*, where memberships of users to communities are modeled with a bipartite graph and links among people stem from shared community affiliations [13];
- we introduce *CKB* - a method for distributed generation of large benchmark networks with realistic properties of social graph and social community structure;
- we make our method particularly suitable for benchmarking community detection algorithms by providing the set of parameters for tuning the most important structural properties: number of nodes, mean node degree, edge probability inside a community, power law exponents for distributions of community size and node-community memberships, etc;
- we introduce simple and efficient distributed algorithms for building user-community affiliation network and linking nodes inside communities;
- we develop and evaluate a distributed implementation of the proposed method capable of producing billion-node random social graphs with reference community structure;
- we make our implementation accessible to the research community by providing it as a web service with possibility to download the generated graphs[1].

The rest of the paper is organized as follows. Section 2 contains problem description and section 3 describes the details of CKB. In section 4 accuracy and performance of the method are evaluated. We conclude in section 5 with possible future directions.

2 Problem

Let's consider a graph $G = (V, E)$, where $|V| = N_1$ and $|E| = m$. A community C_i with $|C_i| = n_{c_i}$ is defined as an induced subgraph. The number of communities is N_2, all communities together constitute a *cover* of a graph. The number of entries of j-th node into different communities (*node-community memberships*) is m_j.

The *internal* (d_j^{int}) and *external* (d_j^{ext}) degree of vertex $j \in C_i$ are defined as the number of edges connecting j to other vertices in C_i or to the rest of the graph respectively. So the *total* degree of vertex j is $d_j = d_j^{int} + d_j^{ext}$. Number of edges inside C_i is $d_{c_i} = \frac{1}{2} \sum_{\forall j \in C_i} d_j^{int}$.

The task is to generate G with the following properties:

1. power law degree distribution: $p(d) \sim d^{-\beta}$ [3];
2. giant connected component presence: $\exists\ G^* \subset G : |V^*| \sim N_1, \forall\ i, j \in V^*$ $\exists\ w_1, w_2, ..., w_k \in E^*\ \exists\ t_l \in V$: $w_0 = (i, t_0), w_1 = (t_0, t_1), ..., w_D = (t_{D-1}, j)$ [11];

[1] `http://ckb.at.ispras.ru/`

3. small effective diameter: $\forall\ i,j \in V^* \exists\ w_1, w_2, ..., w_k \in E^*\ \exists\ t_l \in V : w_0 = (i, t_0), w_1 = (t_0, t_1), ..., w_{D_{i,j}} = (t_{D_{i,j}-1}, j)$ such that $\mathbb{P}\left((1-\epsilon)\frac{\ln N_1}{\ln \ln N_1} \leq D_{i,j} \leq (1+\epsilon)\frac{\ln N_1}{\ln \ln N_1}\right) \to 1$ for $N_1 \to \infty$ [1];
4. users could have zero degrees and memberships: $d_j \geq 0$, $m_j \geq 0$;
5. communities are overlapping: $\exists\ C_i, C_j : C_i \cap C_j \neq \emptyset$ [6];
6. each community C_i is connected with high probability: $\mathbb{P}(C_i \text{ is connected}) \geq 1 - \frac{1}{n_{C_i}}$;
7. intra-community density is larger than the average link density of whole graph G: $\frac{d_{C_i}}{n_{C_i}(n_{C_i}-1)} > \frac{m}{N_1(N_1-1)}$ [7];
8. number of edges inside the community is greater than number of edges linking vertices of the community with the rest of the graph: $d_{C_i} > \sum_{\forall j \in C_i} d_j^{ext}$;
9. number of edges in the community increases superlinearly with the community size: $d_{C_i} \propto n_{C_i}^{1+\gamma}$, where $\gamma \in (0, 1)$ [14];
10. user-community memberships have power-law distribution: $p(m_i) \sim m_i^{-\beta_1}$ [14];
11. size of communities has power-law distribution: $p(n_{C_i}) \sim n_{C_i}^{-\beta_2}$ [5];
12. overlaps of communities are more densely connected than the non-overlapping parts: $\forall\ C_i \forall\ C_j (i \neq j), C_{i \cap j} = C_i \cap C_j \Rightarrow \frac{d_{C_{i \cap j}}}{n_{C_{i \cap j}}(n_{C_{i \cap j}}-1)} > \frac{d_{C_i}}{n_{C_i}(n_{C_i}-1)}$ [13];
13. low-degree nodes tend to be part of very few communities, while high-degree nodes tend to be members of multiple groups: $d_i \sim m_i$ [8].

3 Method

The main steps of CKB graph generator are:

1. Degree sequences for users and communities are generated on the assumption of input parameters;
2. Users are assigned to communities using modified configuration model [9];
3. Edges inside each community are generated using configuration model.

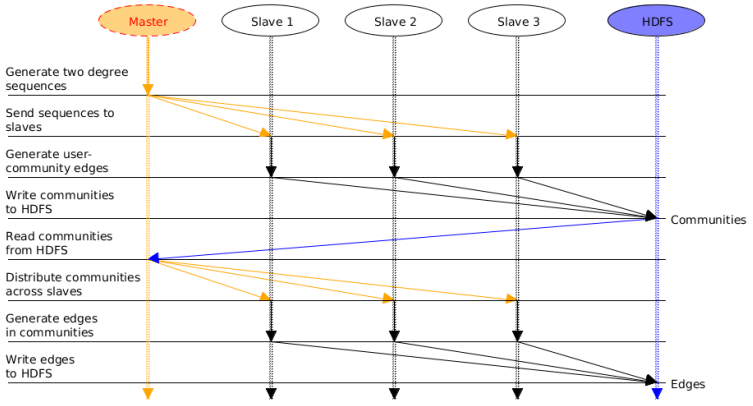

Fig. 1: General workflow

General workflow is shown in Figure 1. *Master node* is the central node of computational cluster and by *slave nodes* we mean the rest of cluster nodes. HDFS files are distributed across local file systems of slave nodes. On the master node, the non-distributed part of computations is carried. During the distributed com-

putations, the master node assigns tasks to slave nodes and aggregates the results. First step of the generation process is done once on the master node while second step is distributed across slave nodes. During the third step edge generation inside each community is performed in a distributed way, so that each slave node generates some part of edges which are then merged.

Table 1: Parameters of CKB

Parameter	**Meaning**	**Default value**
N_1	number of nodes	–
d_{mean}	mean node degree	–
x_{min}	minimum user-community memberships	1
m_{min}	minimum community size	2
x_{max}	maximum user-community memberships	10,000
m_{max}	maximum community size	10,000
$\beta_1 > 1$	power law exponent of user-community membership distribution	2.5
$\beta_2 > 1$	power law exponent of community size distribution	2.5
$\alpha > 0$	affects edge probability inside communities	4
$0 < \gamma < 1$	affects edge probability inside communities	0.5
ϵ	controls the number of edges in ϵ-community	$2N_1^{-1}$

3.1 Users-communities bigraph generation

Bipartite graph (or **bigraph**) is a graph whose vertices can be divided into two disjoint sets U and V and such that every edge connects a vertex in U to one in V. In our case V ($|V| = N_1$) is a set of nodes and U ($|U| = N_2$) is a set of communities. User-community affiliations are modeled as bigraph edges.

1. Number of users (nodes) N_1 is a parameter. At first on the master node number of communities N_2 is computed from the equation:
$$M_0 = N_1 \cdot \mathbb{E}[m] = N_2 \cdot \mathbb{E}[x], \tag{1}$$
where $\mathbb{E}[m]$ and $\mathbb{E}[x]$ are the expectation of node memberships and community sizes respectively. The number of generated edges is defined as
$$M = (1 + \mathbb{E}[\mathbb{P}_{c_i,j}^{mult}])M_0, \tag{2}$$
where $\mathbb{P}_{c_i,j}^{mult}$ is multiple edge probability (section 3.3) which helps to reduce the bias introduced by deleting multiple edges.
k-th moment of the random variable distributed by power law with exponent (β_1 for membership distribution and β_2 for community size distribution are parameters) is
$$\mathbb{E}[x^k] = \int_{x_{min}}^{x_{max}} x^k p(x)dx = \int_{x_{min}}^{x_{max}} x^k \frac{1-\beta}{x_{max}^{1-\beta} - x_{min}^{1-\beta}} x^{-\beta} dx, \tag{3}$$
since $p(x) = \frac{1-\beta}{x_{max}^{1-\beta} - x_{min}^{1-\beta}} x^{-\beta}$. So,
$$\mathbb{E}[x^k] = \frac{(1-\beta)(x_{max}^{k+1-\beta} - x_{min}^{k+1-\beta})}{(x_{max}^{1-\beta} - x_{min}^{1-\beta})(k+1-\beta)} \tag{4}$$
Note that for $k - \beta + 1 = 0$ the expectation equals to
$$\mathbb{E}[x^k] = \frac{1-\beta}{x_{max}^{1-\beta} - x_{min}^{1-\beta}} \ln\left(\frac{x_{max}}{x_{min}}\right)$$

2. Identical power law degree sequences are generated on each slave node.
3. Each vertex is associated with degree (d_i^1 for i-th user-node and d_j^2 for j-th community-node) from degree sequence that was generated at previous step.
4. Numbers $D_1^1 = d_1^1$, $D_2^1 = D_1^1 + d_2^1$, ... , $D_{k+1}^1 = D_k^1 + d_{k+1}^1$, ..., $D_{N_1}^1 = D_{N_1-1}^1 + d_{N_1}^1$ and $D_1^2 = d_1^2$, $D_2^2 = D_1^2 + d_2^2$, ... , $D_{k+1}^2 = D_k^2 + d_{k+1}^2$, ... ,$D_{N_2}^2 = D_{N_2-1}^2 + d_{N_2}^2$ are computed.
5. For the sequence of natural numbers
$$[M] = \{1, 2, 3, \ldots, \lfloor \frac{M}{s} \rfloor\},$$
where $\lfloor x \rfloor = max\{n \in \mathbb{Z} | n \leq z\}$ and s is the number of slave nodes, compute in a loop on each slave node:
for $t = 1$ to $\lfloor \frac{M}{s} \rfloor$ do:
   (a) choose random natural numbers p and q from $[M]$ with uniform distribution;
   (b) find the interval $[D_i^1, D_{i+1}^1]$ to what the number p belongs;
   (c) find the interval $[D_j^2, D_{j+1}^2]$ to what the number q belongs;
   (d) if $i \neq j$ add to the bigraph an edge (i, j).
6. Merge all generated edges and remove multiple edges.

Complexity of this stage is $O(M \log(N_1 N_2))$.

3.2 Intra-community edges generation

At this stage edges between nodes are generated in conformity with their belonging to communities. We sample number of edges in community C_j from Binomial distribution (considering the number of multiple edges):

$$M_{c_j} = \frac{1}{s}(1 + \mathbb{P}_{c_k}^{mult})\text{Bin}(x_{c_k}, p_{c_k}), \tag{5}$$

where x_{c_k} is community size, p_{c_k} is edge probability in the community c_k, s is the number of slave nodes, and $\mathbb{P}_{c_k}^{mult}$ is multiple edge probability (section 3.3) which helps to reduce the bias introduced by deleting multiple edges.

Then, on each slave node M_{c_j} edges are generated using configuration model. Finally, all generated edges are merged and multiple edges are removed. Self-loops are filtered during the generation process.

For each pair of nodes i and j in the community c_k the probability of edge (i, j) is defined as

$$p_{c_k} = \frac{\alpha}{x_{c_k}^{\gamma}}, \tag{6}$$

where α and γ are parameters ($0 < \gamma < 1$, $\alpha > 0$) [14].

The total probability of an edge between i and j in overall graph is

$$p(i, j) = 1 - \prod_{c_k \in C_{ij}} (1 - p_{c_k}), \tag{7}$$

where C_{ij} is a set of communities that i and j share [14].

Therefore, overlaps of communities are more densely connected than the non-overlapping parts. Also low-membership nodes will have low-degree. And vice versa, if membership of node increases then its degree is growing too.

ϵ-community To allow for edges between nodes that don't share any common communities, we add an additional ϵ-community [13] which connects any pair of

nodes with a small probability ϵ. This step is also necessary to ensure the existence of zero membership nodes with non-zero degrees in the resulting graph. In other words, some part of users with low degrees are not members of any community.

The number of edges generated on each slave node is

$$M_\epsilon = \frac{1}{s}\frac{N_1(N_1-1)}{2}\epsilon, \tag{8}$$

where ϵ is a parameter.

Complexity of this stage is $O(K_m)$, where $K_m = \sum_{c_j} M_{c_j}$.

3.3 Multiple edges

Knowing multiple edge probability for each step of generation helps to reduce the bias introduced by deleting multiple edges produced by configuration models. So for **users-communities bigraph generation** the probability that in bipartite graph an edge appears two or more times will be

$$\mathbb{P}^{mult}_{c_i,j} \approx \Big(\frac{x_{c_i} m_j}{M}\Big)^2 \tag{9}$$

For **intra-community edges generation** the probability of multiple edge is

$$\mathbb{P}^{mult}_{c_k} \approx \frac{\alpha^2}{4x^{2\gamma}_{c_k}} \tag{10}$$

3.4 Mean degree

Since mean degree is an important feature for graph analysis and community detection algorithm testing, we obtained the dependence between input parameters α and γ and mean degree. Calculation of the mean degree allows to prove that density of edges inside the community is increased than in overall graph. Due to limited space we provide only the final equations. So the mean degree is:

$$d_{mean} \approx \frac{(S_1 - S_2 + S_3)\binom{N_1}{2}}{N_1}, \tag{11}$$

where N_1 is the number of nodes in the whole graph and S_1, S_2 and S_3 are

$$S_r = \alpha^r \mathbb{E}\Big[\sum_{c_1<\ldots<c_r}\prod_{k=\{1,\ldots,r\}}\frac{1}{x^\gamma_{c_k}}\prod_{t=\{i,j\}}\Big(\frac{x_{c_k}m_t}{M}\Big)\Big] \tag{12}$$

Each moment $\mathbb{E}[x^y]$ can be computed from (4). Now after solving the cubic equation in variable α (and fixed γ) we can compute probability p_{c_i}. But the (11) is valid only with some constrains, that are not provided due to limited space.

3.5 Connectedness of community

Using known results on evolution of random graphs in Erdos-Renyi model [2] we can claim the following:

Theorem 1 *The community C_i is connected with high probability for*

$$\alpha > \ln(x_{c_i})x^{\gamma-1}_{c_i} \tag{13}$$

4 Evaluation

We implemented the proposed method in Scala using Apache Spark[2] - a framework for efficient computations in distributed environment.

The results of running time evaluation on Amazon EC2 clusters and single machine are shown in Figure 2. Near-linear scalability on the number of nodes in

[2] http://spark.incubator.apache.org/

the generated graph allows to produce synthetic networks of huge size in reasonable time: one billion nodes graph generation took < 2 hours on Amazon EC2 cluster with 100 *m1.large* instances.

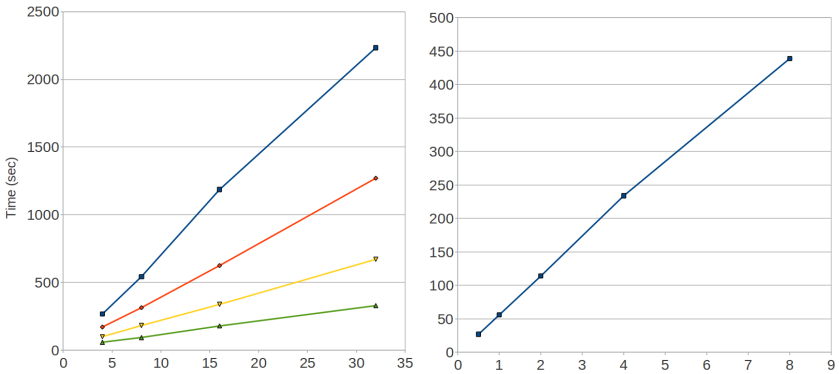

Fig. 2: Scalability evaluation results. **Left: Amazon EC2 clusters of *m1.large* instances**, blue line - 2 slave nodes, red line - 4 slave nodes, yellow line - 8 slave nodes, green line - 16 slave nodes. **Right: single machine**.

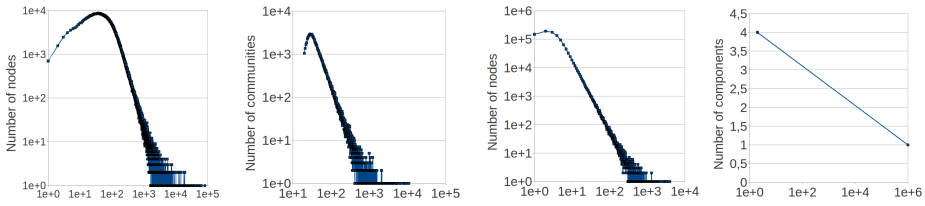

Fig. 3: Community size, user-community memberships, degree distribution and connected component distribution for $N_1 = 10^6$, $\beta_1 = \beta_2 = 2.5$

Table 2: Comparison of CKB against SNAP networks and LFR

	Orkut	**LiveJournal**	**YouTube**	**CKB**		**LFR**
Number of nodes	3M	4M	1.1M	3M	97.5K	100K
Mean degree	76.2	17.3	5.3	109.9	68.8	66.7
Community size power law exponent $\beta_{comm_{size}}$	2.12	2.14	2.36	2.19	2.57	2.54
Membership power law exponent β_{memb}	1.59	2.22	2.83	2.28	2.62	–
Degree distribution power law exponent β_{graph}	1.58	2.15	2.53	2.22	2.54	2.56
Community size distribution median	16	2	3	5	49	40
Membership distribution median	14	2	1	7	1	1
Average clustering coefficient	0.169	0.353	0.172	0.039	0.055	0.226
Effective diameter d_{eff}	4.8	6.4	6.5	4.38	3.88	3.98
Generation time (sec)	–	–	–	160	11	863

Table 2 summarizes the most important statistics of LiveJournal, ORKUT and YouTube datasets from *Stanford Large Network Dataset Collection*[3] and compares them with CKB and LFR benchmarks [5]. Comparing the tables suggests that CKB graphs have very similar structural properties to real networks. The only difference is low average clustering coefficient of the generated networks. However, achieving more realistic clustering coefficient requires some changes in the edge generation process and is a subject of future work.

5 Conclusion

A method for distributed generation of large benchmark networks with realistic properties of social graph and social community structure has been introduced and evaluated. Possible directions for future work include:

- distributed computation of Normalized Mutual Information or other measures for comparing covers of communities;
- testing different community detection algorithms;
- allow to control clustering coefficient and degree correlation of nodes.

References

1. Albert, R., Jeong, H., and Barabasi, A.-L. Diameter of the world wide web. *Nature 401* (1999), 130–131.
2. Erdos, P., and Renyi, A. On the evolution of random graphs. *Bull. Inst. Int. Statist. Tokyo 38* (1961), 343–347.
3. Faloutsos, M., Faloutsos, P., and Faloutsos, C. On power-law relationships of the internet topology. *SIGCOMM* (1999), 251262.
4. Fortunato, S. Community detection in graphs. *Physics Reports 486*, 3 (2010), 75–174.
5. Lancichinetti, A., and Fortunato, S. Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities. *Phys. Rev. 80(1)* (2009).
6. Lancichinetti, A., Fortunato, S., and Kertsz, J. Detecting the overlapping and hierarchical community structure in complex networks. *New J. Phys.* (2009).
7. Leskovec, J., Lang, K. J., Dasgupta, A., and Mahoney, M. Statistical properties of community structure in large social and information networks.
8. Mislove, A., Marcon, M., Gummadi, K., Druschel, P., and Bhattacharjee, B. Measurement and analysis of online social networks. *IMC* (2007).
9. Molloy, M., and Reed, B. A critical point for random graphs with a given degree sequence. *Random Structures and Algorithms 6* (1995), 161–180.
10. Plantié, M., and Crampes, M. Survey on social community detection. In *Social Media Retrieval*. Springer, 2013, pp. 65–85.
11. Spencer, J. The giant component: The golden anniversary. *Not. Am. Math. Soc. 401* (1999), 130–131.
12. Xie, J., Kelley, S., and Szymanski, B. K. Overlapping community detection in networks: the state of the art and comparative study. *arXiv preprint arXiv:1110.5813* (2011).
13. Yang, J., and Leskovec, J. Community-afliation graph model for overlapping network community detection. In *IEEE 12th International Conference on Data Mining* (2012).
14. Yang, J., and Leskovec, J. Structure and overlaps of communities in networks. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (2012).

[3] http://snap.stanford.edu/data/index.html