SAND2018-5016C

Scalable Community Detection Benchmark Generation*

Jonathan Berry† Cynthia Phillips‡ Siva Rajamanickam§ George M. Slota¶

Abstract

We introduce wrapped BTER, a way to take a single input to the Block Two-Level Erdős-Rényi (BTER) [4] scalable graph generator and produce a family of related graphs which vary by tightness of connection within BTER affinity blocks. This gives an alternative to the Lancichinetti-Fortunato-Radicchi (LFR) community detection benchmark capable of generating billion-edge graphs in less than a minute.

1 Introduction

A Google Scholar search for "community detection" returns 3,650 results for 2017 alone. These are papers that add algorithmic capabilities or use community detection for an application. See [2] for a survey of some of the community-detection algorithms developed by 2010. The de facto standard method to compare community-detection algorithms uses graph instances generated by the LFR benchmark [5]. LFR tries to match a given degree distribution and community-size distribution. It also provides a parameter μ that controls the "tightness" of its ground-truth communities. Specifically, μ is the average over all vertices of the fraction of inter-community edges. LFR performs an expensive iterative "rewiring" to approximately meet the community tightness goal. While LFR is a useful benchmark, the best parallel implementation requires 17 hours on 16 threads to produce a graph with 1 billion edges [3].

The BTER generator tries to match two graph properties important for realistic community structure: the degree distribution and the per-degree clustering coefficients. The clustering coefficient (CC) for a vertex is the fraction of its wedges that close into triangles. A wedge for vertex v with neighbors u and w is the length-2 path $(u - v - w)$. The per-degree CC c_d for degree d is the average CC over all nodes of degree d. This input to BTER can come from goal distributions or from real graphs a researcher wishes to mimic. Since BTER does not try to meet a goal μ, its edge-generation process is more direct, simpler,and faster.

Our goal is to use BTER to provide an approximation to LFR that enables more practical evaluation of community-detection algorithms on high-performance computing (HPC) resources. Given one set of BTER inputs, we derive a family of BTER inputs each of which produces graphs that approximately meet a community-tightness goal. Thus, we use BTER as a black box.

2 wBTER

We generate graphs with two steps. Given a goal parameter μ_g similar to LFR's μ and the two distributions BTER requires, we run a linear program (LP) to obtain a new CC distribution to feed BTER. Our new CC distribution $\hat{c}_d$ constrains BTER to generate graphs with the desired μ_g.

Our parameter μ_g is a relaxation of the LFR's μ parameter. Rather than averaging over all vertices, we use the fraction of inter-community edges in the whole graph, which we call the *global tightness*. Unlike LFR, BTER instances have tiny affinity blocks and singleton vertices not in any block. The strict LFR μ a poor fit for this context. Our relaxation retains the spirit of LFR for all but the smallest affinity blocks.

We treat BTER's affinity blocks (ABs) as ground truth communities. ABs normally have $d + 1$ vertices of degree d. BTER builds an Erdős-Rényi graph on each AB with minimum degree d, using probability $p_d = \sqrt[3]{c_d}$. This original CC distribution induces an expected global tightness μ_B.

Our LP is defined below. We quantify the per-degree deviations between the AB edge probability and the original global tightness, then minimize the differences in these deviations between the c_d and $\hat{c}_d$ clustering coefficients subject to the constraint that the resulting global tightness is μ_g. Define $p_B = (1 - \mu_B)$, which is the global edge probability in the native BTER instance. Let p_d be the original AB edge probability for degree-d nodes. The expected number of intra-block

*Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energys National Nuclear Security Administration under contract DE-NA-0003525. The research presented in this paper was funded through the Laboratory Directed Research and Development (LDRD) program at Sandia National Laboratories, in the context of the Multi-Level Memory Algorithmics for Large, Sparse Problems Project.

†Sandia National Labs, NM, jberry@sandia.gov
‡Sandia National Labs, NM, caphill@sandia.gov
§Sandia National Labs, NM, srajama@sandia.gov
¶Rensselaer Polytechnic Institute, NY, slotag@rpi.edu

edges given the original c_d distribution is:

$$\sum_d dn_d p_d = 2m(1-\mu_B) = 2mp_B$$

where n_d is the number of vertices with degree d and m is the total number of edges. We achieve the goal μ_g by defining variables $\hat{p}_d$ for new AB probabilities. Our primary constraint is:

$$\sum_d dn_d \hat{p}_d = 2m(1-\mu_g)$$

To obtain our objective function, we define degree-wise local edge *surpluses* for the original and target BTER parameters, and a global surplus, respectively:

$$\begin{aligned} s_d &= dn_d p_d - dn_d p_B \\ \hat{s}_d &= dn_d \hat{p}_d - dn_d p_B \\ s_g &= 2m(1-\mu_g) - 2m(1-\mu_B). \end{aligned}$$

Our objective is to minimize the absolute difference between local and global surpluses. Our LP is:

$$\begin{aligned} \text{minimize} \quad & \sum_d |\hat{s}_d - s_d| \\ \text{subject to} \quad & \sum_d \hat{s}_d - \textstyle\sum_d s_d = s_g \\ & \sum_d dn_d \hat{p}_d = 2m(1-\mu_g) \\ & 0 \le \hat{p}_d \le 1 \end{aligned}$$

We also weight the objective to resist changes in the tail of the CC distribution, include smoothing constraints, and omit degree-1 vertices (not shown). We compute new CC's from the LP solution: $\hat{c}_d = \hat{p}_d^3$. We then feed the original degree distribution and the new $\hat{c}_d$ to BTER. We call this process *wrapped BTER* or wBTER.

3 Results

We first ensure that we can achieve μ_g when mimicking a real graph. We use a variation of the LiveJournal social network crawl from the SNAP Datasets [1]. The graph has about 4 million vertices and 26 million undirected edges. In Figure 1 (left) we show our target μ_g versus the empirical global tightness μ of generated instances. We achieve our target with minimal error in all instances.

We evaluate performance on a 68-core Intel Knight's Landing (KNL) processor in *cache mode*, We use the same distributions and varying $\mu \in \{0.1, \ldots, 0.6\}$. Figure 1 gives the performance in MEPS (millions of edges processed per second). Performance increases with μ, a consequence of the coupon collectors process BTER uses [4]. Our LP runs take less than 1s.

[1] http://snap.stanford.edu/data

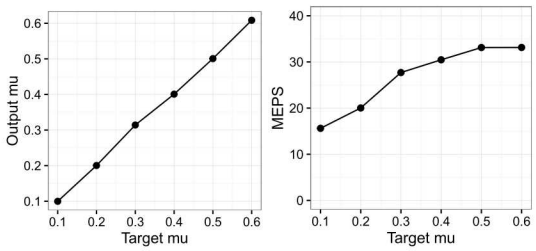

Figure 1: (**Left**) Target μ versus output μ. (**Right**) MEPS versus μ.

In preliminary scaling experiments with synthetically-generated distributions[2], we create graphs with 100M vertices and 2B edges in under one minute on KNL. We compare this performance to the original LFR code[3], which takes over 24 hours to produce a graph with 1M vertices and 16M edges on an Intel Core i5-5287U. A parallel LFR implementation by Hamann et al. [3] running with 16 threads on an Intel Xeon E5-2630 v3 requires about 17 hours to generate a 100M vertex 1B edge instance with $\mu = 0.2$.

4 Ongoing and Future Work

These encouraging preliminary results motivate several research directions. The wBTER approach must be tested on large datasets such as Friendster (1.8B edges) to further prove its scalability. We believe that it can be adapted to generate instances representative of real social networks varying not only global tightness, but graph size. We intend to study the network properties of wBTER instances as we vary μ_g. The wBTER can be further optimized for high-performance architectures, and it remains to show that LFR-style comparisons of community detection algorithms are reliable.

References

[1] W. Aiello, F. Chung, and L. Lu, *A random graph model for power law graphs*, Experimental Mathematics, 10 (2001), pp. 53–66.

[2] S. Fortunato, *Community detection in graphs*, Physics reports, 486 (2010), pp. 75–174.

[3] M. Hamann, U. Meyer, M. Penschuck, and D. Wagner, *I/o-efficient generation of massive graphs following the lfr benchmark*, in 2017 Proceedings of the Ninteenth Workshop on Algorithm Engineering and Experiments (ALENEX), SIAM, 2017, pp. 58–72.

[4] T. G. Kolda, A. Pinar, T. Plantenga, and C. Seshadhri, *A scalable generative graph model with community structure*, SIAM Journal on Scientific Computing, 36 (2014), pp. C424–C452.

[2] http://www.sandia.gov/~tgkolda/feastpack/doc_bter_ideal.html

[3] https://sites.google.com/site/santofortunato/inthepress2

[5] A. Lancichinetti, S. Fortunato, and F. Radicchi, *Benchmark graphs for testing community detection algorithms*, Physical Review E, 78 (2008), pp. 1–5.

[6] J. Leskovec and A. Krevl, *SNAP Datasets: Stanford large network dataset collection*. `http://snap.stanford.edu/data`, June 2014.