ARTIFICIAL BENCHMARK FOR COMMUNITY DETECTION (ABCD)

Bogumił Kamiński, Paweł Prałat and François Théberge

SIAM Workshop on Network Science 2020
July 9–10 · Toronto

Summary

We propose a new method for generating graphs with communities that is not only faster but also has a more natural interpretation than the current state of the art.

Introduction

The standard and extensively used method for generating artificial networks is the **LFR** graph generator [6]. This model has some scalability limitations and it is challenging to analyze it theoretically. Moreover, the mixing parameter μ, the main parameter of the model guiding the strength of the communities, has a non-obvious interpretation and so can lead to unnaturally-defined networks.

We provide an alternative random graph model with community structure and power-law distribution for both degrees and community sizes, the **A**rtificial **B**enchmark for **C**ommunity **D**etection (**ABCD** graph). We show that the new model solves the three issues identified above and more. Indeed, it is fast, simple, and can be easily tuned to allow the user to make a smooth transition between the two extremes: pure (independent) communities and random graph with no community structure. We illustrate the latter in Figure 1, where all graphs have the same degree distribution and community sizes. The three graphs correspond to increasing values of the mixing parameter μ (for **LFR**) or ξ (for **ABCD**). Edges that fall between vertices in the same community are coloured accordingly. We see strong communities for the leftmost plots, and noisy yet still coherent communities for the middle plots. The rightmost plots illustrate our point regarding one of the main differences between **LFR** and **ABCD**. For **LFR**, in the top right plot, we see almost no edges within each community so the model generates "anti-communities". With **ABCD**, we see a random looking graph, where the number of edges within each community is proportional to the number of vertices that belong to it, as expected in a random graph.

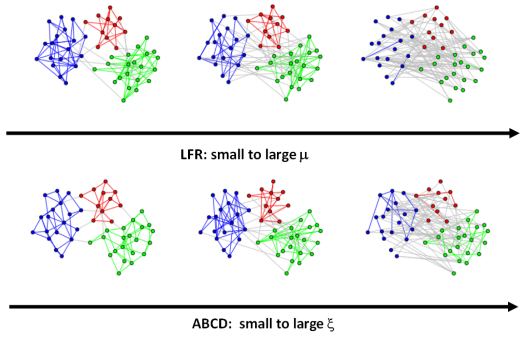

Figure 1: Examples of graphs generated by the **LFR** model (top) and by the **ABCD** model (bottom).

ABCD Models

We briefly discuss the different flavours of the **ABCD** benchmark—full details can be found in [4]. As with **LFR**, for a given number of vertices n, we start by generating a power law distribution both for the degrees and community sizes. Those are governed by the power law exponent parameters (γ, β). We also provide extra information to the model, again as with **LFR**, namely, the average and maximum degree, and the range for the community sizes.

For each community, we generate a random *community* subgraph using either the Configuration Model (CM, see[2]) which preserves the exact degree distribution, or the Chung-Lu model (CL, see [3]) which preserves the expected degree distribution. We also generate a *background* random graph with the same degree distribution. The mixing parameter ξ guides the proportion of edges which are generated via the background graph. In particular, when $\xi = 1$, the graph has no community structure while with $\xi = 0$ we get disjoint communities. In order to generate simple graphs, we may have to do some re-sampling or edge re-wiring, which are described in [4]. This two-step process is similar to the highly scalable **BTER** model [5].

With this process, larger communities will get slightly more internal edges (in proportion) due to the background graph. In order to provide a variant where the expected

proportion of internal edges is the same for every community (as with **LFR**), we also provide a "local" version of **ABCD** where the mixing parameter ξ is adjusted for every community.

Performance

We compare efficiency of the generating algorithms. All the results were obtained on a single thread of Intel Core i7-8550U CPU @ 1.80GHz, run under Microsoft Windows 10 Pro, and performing all computations in RAM. The computations for **LFR** were performed using the C++ language implementation[1]. For **ABCD**, the Julia 1.3 language implementation was used [1] in order to ensure high performance of graph generation, while keeping the size of the code base small. We tested all four combinations of the **ABCD** model (Chung-Lu vs. Configuration Model, and global vs. "local" ξ's). We show some results in Figure 2 where we vary the number of vertices from under 10,000 to 500,000. We see a roughly 100-fold speedup with the **ABCD** models.

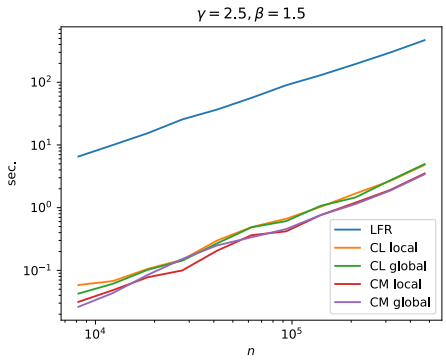

Figure 2: Generation times in seconds of the **LFR** and the **ABCD** models.

Properties

Next, we compare graphs generated with the **LFR** and the **ABCD** benchmarks via some graph statistics: clustering coefficient (the average vertex transitivity), eigenvector centrality, the global transitivity, and the average shortest paths length (approximated via sampling). We generated graphs with 100,000 vertices, average degree 25, maximum degree 500 and power law exponent $\gamma = 2.5$; for the community sizes, we used power law exponent $\beta = 1.5$ with sizes between 50 and 2000. The mixing parameter for **LFR** is set to $\mu = 0.2$ and, in order to compare similar graphs, for the **ABCD** algorithm we derive the corresponding ξ.

In Figure 3, we report the distribution of the graph properties obtained by generating 30 graphs each using **LFR** as well as 4 variations of **ABCD**: CM and CL respectively with the (g)lobal or (l)ocal ξ's. The results of these experiments show high similarity of the generated graphs, in particular, when the configuration model is used. Indeed, some graph parameters that are sensitive with respect to the degree distribution (such as clustering coefficient) are not as well preserved for the Chung-Lu variant of the model, which is natural and should be expected. Having said that, all graph parameters we evaluated are relatively well aligned.

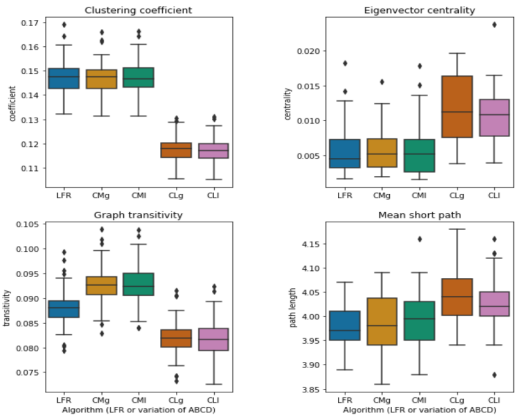

Figure 3: Comparing properties of **LFR** and **ABCD** graphs.

[1] `github.com/eXascaleInfolab/LFR-Benchmark_UndirWeightOvp`

References

[1] J. Bezanson, A. Edelman, S. Karpinski, and V. Shah. Julia: A fresh approach to numerical computing. *SIAM Review*, 69:65–98, 2017.

[2] B. Bollobás. A probabilistic proof of an asymptotic formula for the number of labelled regular graphs. *European Journal of Combinatorics*, 1:311–316, 1980.

[3] F. Chung and L. Lu. *Complex Graphs and Networks*. American Mathematical Society, 2006.

[4] B. Kamiński, P. Prałat, and F. Théberge. Artificial benchmark for community detection (abcd): Fast random graph model with community structure. *pre-print, arXiv:2002.00843*, 2020.

[5] T. G. Kolda, A. Pinar, T. Plantenga, and C. Seshadhri. A scalable generative graph model with community structure. *SIAM Journal on Scientific Computing*, 36(5):C424–C452, 2014.

[6] A. Lancichinetti, S. Fortunato, and F. Radicchi. Benchmark graphs for testing community detection algorithms. *Physical Review E*, 78, 2008.