On the Multi-Dimensional Acceleration of Stochastic Blockmodeling for Community Detection

Frank Wanye
Dept. of Computer Science
Virginia Tech
Blacksburg, USA
wanyef@vt.edu

Wu-chun Feng
Dept. of Computer Science
Virginia Tech
Blacksburg, USA
wfeng@vt.edu

***Abstract*—Stochastic block partitioning (SBP) is a community detection algorithm that is highly accurate even on graphs with a complex community structure. However, SBP is much slower than more commonly used algorithms, such as Louvain, making SBP impractical for analyzing large real-world graphs with millions of edges. Thus, we aim to realize fast and accurate community detection on large graphs by accelerating the highly accurate SBP algorithm via sampling, parallel and distributed computing on a cluster as well as algorithmic optimization. We compare our approach to other community detection algorithms, showing that SBP accelerated with our methods on 64 compute nodes is up to 1,163× faster than the official "Graph Challenge" baseline SBP implementation, while still being more accurate than the Louvain and Leiden algorithms on large graphs.**

***Index Terms*—community detection, graph analytics, stochastic blockmodels, stochastic block partitioning, graph clustering**

I. Introduction

Stochastic block partitioning (SBP) [1] is a community detection algorithm based on inference over the degree-corrected stochastic blockmodel (DCSBM). It is based on a Markov chain Monte-Carlo inference method that minimizes the description length of the DCSBM. SBP is more accurate and less prone to overfitting than the more commonly used modularity-based methods. However, SBP is both slow and based on the inherently sequential Markov chain Monte-Carlo (MCMC) inference, making it hard to parallelize and thus limiting its applicability to real-world graphs, which often have upwards of millions of edges. Table I shows that while SBP is more accurate than the Leiden community detection algorithm, it is 414× to 15,476× slower than SBP.

We present our work on accelerating SBP and making it a more viable option for performing accurate community detection on large graphs. We accelerate SBP through sampling, parallel and distributed computing, and algorithmic optimizations. We then compare our results against two SBP implementations as well as the Louvain [2] and Leiden [3] algorithms built on top of the igraph library [4]. By combining all three acceleration methods we reduce the runtime difference between SBP and Leiden on the 50k vertex graph from 15,476× down to 4.5×, while still being more accurate than Leiden.

This project was supported in part by NSF I/UCRC CNS-1822080 via the NSF Center for Space, High-performance, and Resilient Computing (SHREC).

II. Methods

Below we present our three strategies for accelerating SBP.

A. Data Reduction via Sampling

We develop a sampling framework called SamBaS [5], [6] that (a) preserves community structure, (b) speeds up SBP, (c) generates community detection on the entire graph, not just the sampled portion, and (d) maintains, and in some cases, even improves community detection accuracy over SBP without sampling, given the right sampling parameters.

The method consists of four steps. In Step 1, a subgraph is sampled using one of several vertex sampling algorithms. Then, in Step 2, SBP is run on the sampled subgraph. In Step 3, the community detection results from the sampled subgraph are propagated to the rest of the graph based on connectivity. Finally, in Step 4, the results are fine-tuned by running the Metropolis-Hastings algorithm on the entire graph. We find that the choice of sampling algorithm is greatly influenced by the density and degree distribution of the graph.

B. Parallel and Distributed Computing

The asynchronous Gibbs algorithm [7] has been shown to be an effective means of parallelizing MCMC methods when the number of computational dependencies is low. However, when applied to community detection, it often leads to a drastic decrease in accuracy [8]. To that end, we develop a lock-free hybrid shared-memory parallel algorithm [8]. It estimates the information content [9] of a vertex based on its degree and processes those vertices sequentially using Metropolis-Hastings. Then, it processes the remaining vertices using asynchronous Gibbs. Because real-world graphs usually have a power-law degree distribution, the majority of vertices are processed in parallel, accelerating the computation while maintaining accuracy.

We then develop a distributed-memory parallel algorithm that distributes our parallel SBP implementation across the nodes of a cluster [10]. In this method, we duplicate the graph data across all compute nodes, run the hybrid parallel approach on distinct sets of vertices on each node, and utilize MPI all-to-all communication primitives to synchronize the changes to the blockmodel at the end of each iteration. To ensure load balancing and an even distribution of highly informative

vertices across compute nodes, we assign vertices to compute nodes based on their degrees.

C. Algorithmic Optimizations

We translate the "Graph Challenge" [1] SBP implementation from Python to C++ and then develop algorithmic optimizations to (a) improve the runtime of SBP and (b) improve the accuracy of SBP when used in conjunction with parallel and distributed computing methods.

The runtime optimizations include (a) utilizing disjoint sets to perform block merges in approximately linear time, (b) using a sparse vector of changes to the blockmodel to compute changes in description length and update the blockmodel, (c) storing a cache of precomputed `log` values, and (d) storing the blockmodel matrix transpose for faster column-wise indexing.

Our first accuracy optimization is vertex-level batching to increase the communication frequency in parallel and distributed SBP implementations, leading to less stale information being used in internal computations. The second involves a more accurate way to estimate the information content of a vertex for the hybrid parallel algorithm. Instead of estimating this value based on vertex degree as done in [8], we base this estimate on the product of the degrees of the vertices that make up the edges of the graph. This method is more consistent with the findings in [9].

III. Results

We combine all three acceleration methods described above and compare the resulting accelerated SBP against five of the official "Graph Challenge" graphs with high overlap and high block-size variation. We run our experiments on a 64-node cluster equipped with 128-core AMD EPYC 7702 CPUs and 256 GB of RAM.

Table I summarizes our results in terms of both normalized mutual information and the number of identified communities; SBP is consistently more accurate than Louvain and Leiden. We also show that our accelerated SBP is up to 1,163× faster than the "Graph Challenge" baseline. Thus, we reduce the runtime difference between SBP and Leiden from 1,436× to just 4.5× while still being more accurate than the latter on larger graphs.

IV. Conclusion

We present our work toward accelerating highly accurate community detection. We accelerate SBP - a highly accurate community detection algorithm based on MCMC inference - via sampling, parallel and distributed computation, and algorithmic refinements. Used in combination, our methods accelerate SBP by up to 1163× while maintaining an accuracy advantage over Louvain and Leiden.

In future work, we plan to more thoroughly evaluate the accuracy of SBP against other community detection methods on a wide variety of synthetic and real-world graphs. We also plan to improve the scalability of our approach by introducing data distribution to our distributed SBP method and exploring additional algorithmic refinements.

TABLE I
Results on selected Graph Challenge graphs

Num. Vertices	1k	5k	50k	200k	1M
Algorithm	Normalized Mutual Information				
SBP (Graph Challenge)	0.79	0.87	0.92	-	-
SBP (graph-tool [11])	**0.91**	**0.94**	**0.98**	**0.93**	-
SBP (accelerated)	0.75	0.78	0.90	0.88	**0.83**
Louvain (igraph)	0.78	0.70	0.85	0.82	0.62
Leiden (igraph)	0.82	0.76	0.81	0.80	0.62
Algorithm	Community Ratio[1]				
SBP (Graph Challenge)	0.64	0.58	0.59	-	-
SBP (graph-tool)	**0.73**	**0.74**	**1.22**	**0.59**	-
SBP (accelerated)	0.45	0.37	0.59	0.42	**0.26**
Louvain (igraph)	0.55	0.37	0.43	0.35	0.06
Leiden (igraph)	0.64	0.53	0.39	0.42	0.20
Algorithm	Runtime (s)				
SBP (Graph Challenge)	67	662	30952	-	-
SBP (graph-tool)	5	18	602	6224	-
SBP (accelerated)	4	6	25	117	713
Louvain (igraph)	< 1	< 1	4	47	297
Leiden (igraph)	$< \mathbf{1}$	$< \mathbf{1}$	**2**	**15**	**119**

[1] Community Ratio = Number of communities identified by algorithm divided by the number of communities in ground truth. The closer to 1.0, the better.

References

[1] E. Kao, V. Gadepally, M. Hurley, M. Jones, J. Kepner, S. Mohindra, P. Monticciolo, A. Reuther, S. Samsi, W. Song, D. Staheli, and S. Smith, "Streaming graph challenge: Stochastic block partition," in *2017 IEEE High Performance Extreme Computing Conference (HPEC)*. Waltham, MA: IEEE, 9 2017, pp. 1–12. [Online]. Available: doi.org/10.1109/HPEC.2017.8091040

[2] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2008, no. 10, p. P10008, 10 2008. [Online]. Available: doi.org/10.1088/1742-5468/2008/10/P10008

[3] V. A. Traag, L. Waltman, and N. J. van Eck, "From Louvain to Leiden: guaranteeing well-connected communities," *Scientific Reports 2019 9:1*, vol. 9, no. 1, pp. 1–12, 3 2019. [Online]. Available: doi.org/10.1038/S41598-019-41695-Z

[4] G. Csárdi and T. Nepusz, "The igraph software package for complex network research," *InterJournal*, vol. Complex Systems, p. 1695, 2006. [Online]. Available: https://igraph.org

[5] F. Wanye, V. Gleyzer, and W.-c. Feng, "Fast Stochastic Block Partitioning via Sampling," in *2019 IEEE High Performance Extreme Computing Conference (HPEC)*. Waltham, MA, USA: IEEE, 9 2019, pp. 1–7. [Online]. Available: doi.org/10.1109/HPEC.2019.8916542

[6] F. Wanye, V. Gleyzer, E. Kao, and W.-c. Feng, "SamBaS: Sampling-Based Stochastic Block Partitioning," *arXiv*, 8 2021. [Online]. Available: doi.org/10.48550/arXiv.2108.06651

[7] A. Terenin, D. Simpson, and D. Draper, "Asynchronous Gibbs Sampling," in *International Conference on Artificial Intelligence and Statistics*. Palermo: PMLR, 6 2020, pp. 144–154. [Online]. Available: doi.org/10.48550/arXiv.1509.08999

[8] F. Wanye, V. Gleyzer, E. Kao, and W.-c. Feng, "On the Parallelization of MCMC for Community Detection," in *Proceedings of the 51st International Conference on Parallel Processing*. New York, NY, USA: ACM, 2022, pp. 1–13. [Online]. Available: doi.org/10.1145/3545008.3545058

[9] E. K. Kao, S. T. Smith, and E. M. Airoldi, "Hybrid Mixed-Membership Blockmodel for Inference on Realistic Network Interactions," *IEEE Transactions on Network Science and Engineering*, vol. 6, no. 3, pp. 336–350, 7 2019. [Online]. Available: doi.org/10.1109/TNSE.2018.2823324

[10] F. Wanye, V. Gleyzer, E. Kao, and W.-c. Feng, "Exact Distributed Stochastic Block Partitioning," *arXiv*, 5 2023. [Online]. Available: doi.org/10.48550/arXiv.2305.18663

[11] T. P. Peixoto, "The graph-tool python library," *figshare*, 2014. [Online]. Available: doi.org/10.6084/m9.figshare.1164194