SAND2015-3901C

Improving the Performance of Graph Analysis Through Partitioning with Sampling

Michael M. Wolf*
Sandia National Laboratories
Albuquerque, NM 87185

Benjamin A. Miller†
Massachusetts Institute of Technology
Cambridge, MA 02139

***Abstract*—Numerous applications focus on the analysis of entities and the connections between them, and such data are naturally represented as graphs. In particular, the detection of a small subset of vertices with anomalous coordinated connectivity is of broad interest, for problems such as detecting strange traffic in a computer network or unknown communities in a social network. Eigenspace analysis of large-scale graphs is useful for dimensionality reduction of these large, noisy data sets into a more tractable analysis problem. When performing this sort of analysis across many parallel processes, the data partitioning scheme may have a significant impact on the overall running time. Previous work demonstrated that partitioning based on a sampled subset of edges still yields a substantial improvement in running time. In this work, we study this further, exploring how different sampling strategies, graph community structure, and the vertex degree distribution affect the partitioning quality. We show that sampling is an effective technique when partitioning for data analytics problems with community-like structure.**

I. Introduction/Background

A. Graph Analytics and Signal Processing for Graphs

Due to their utility in identifying subtle patterns of coordination, graph analytics have proven useful in several diverse application areas. Graph analysis—analysis of entities and the connections or interactions between them—has been used in ISR (Intelligence, Surveillance, and Reconnaissance), where graphs represent entities and relationships detected through multiple sources), social networks (where graphs represent relationships between individuals or document), and cyber security (where graphs may represent communication patterns of computers on a network). The objective of the graph analysis in each of these applications is to find anomalous subgraphs in the data, i.e., subsets of the entities with a statistically unlikely connection pattern. The application of graph analytics to these areas becomes harder as the patterns of coordination grow more subtle and the background graph grows larger and noisier.

A common way of representing a graph is using an adjacency matrix. For an unweighted graph $G = (V, E)$, where V is the set of vertices and E is the set of edges, an adjacency matrix $A = \{a_{ij}\}$ is 1 if there is an edge from vertex i to vertex j (where the vertices are given arbitrary numeric labels) and is 0 elsewhere. This matrix representation enables the use of classical linear algebra tools for graph analysis.

A recent statistical framework has been developed for the detection of subtle patterns in massive datasets as part of an effort called Signal Processing for Graphs (SPG) [1]. The goal of this project is to apply classical signal processing concepts to graph data in order to find these anomalies. The detection algorithms in the SPG framework are based on the concept of a residual graph, which is formed by subtracting the expected graph data from the observed graph data, using the graph's matrix representation. The detection framework is designed to detect coordinated deviations from the expected graph topology (coordination in the residual graph). The SPG processing chain has several stages including temporal integration (where the graph is formed from multiple time steps), construction of the expected topology graph, dimensionality reduction, detection of anomalous subgraphs, and identification of those anomalous subgraphs (assuming such a subgraph is detected).

Dimensionality reduction is a particularly important step in the processing chain, as it makes the detection and identification procedures feasible. Typically, this is done by decomposing the residual matrix through partial eigendecomposition (although a singular value decomposition would work as well). Although relatively strong signals can be detected with only one eigenvector, the more powerful detection methods that are needed to detect more subtle anomalies require hundreds of eigenvectors. This dimensionality reduction step dominates the computation in the graph analytic processing chain, with the sparse matrix-dense vector kernel (SpMV) being the most time consuming kernel. Thus, our efforts in this paper and our prior work have been to improve the performance of this dimensionality reduction step, focusing on speeding up the solution of the eigensystem, $Bx_i = \lambda_i x_i$, where $B = A - \mathbb{E}[A]$ is the residual matrix.

B. Eigenspace Analysis of Large Networks

Computing extreme eigenvalues (and the associated eigenvectors) of matrices based on graphs is a fundamental problem in a wide variety of application areas. Common methods for community detection include eigenvector analysis of the graph Laplacian [2] or the modularity matrix [3]. A common measure of vertex importance is PageRank [4], which is based

*Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000

Dr. Wolf is with the Scalable Algorithms Department at Sandia (email: mmwolf@sandia.gov).

†Mr. Miller is with Lincoln Laboratory at MIT (email: bamiller@ll.mit.edu).

on the principal eigenvector of a modified adjacency matrix. Eigenvector analysis is also frequently used in anomaly detection (e.g., [1]). In all of these applications, a sparse eigensolver is required to efficiently perform the desired computation.

Eigenspace analysis of extremely large graphs requires parallel processing, and in this context significant complications arise. Partitioning data across several processes when the data are in the form of a social or computer network is a relatively new problem. Traditional matrix partitioning algorithms have, for the most part, focused on and been most effective for sparse matrices that have a regular structure (e.g., matrices derived from meshes). However, there has been recent research focused on partitioning large networks—with community structure and skewed degree distributions, as in many graphs of interest—to improve the time required to multiply a sparse matrix by a dense vector (denoted SpMV), which is the key kernel for a sparse eigensolver, often dominating the run time of the solver. In this paper, we focus on sparse matrix partitioning to optimize this SpMV operation.

C. Data Partitioning for Large, Scale-Free Graphs

Matrix partitioning for SpMV operations has traditionally focused on 1D distributions, where a set of rows is assigned to a process, although there has been more recent work on 2D distributions (e.g., [5]). For problems with low data locality (such as networks), a 1D distribution may require all processes to communicate to each other. Yoo et al. showed that randomly partitioning in two dimensions provides a substantial speedup over 1D partitioning [6] for scale-free graphs when a 2D block Cartesian structure is imposed that bounds the number of messages communicated to at most $2(\sqrt{P}-1)$ instead of $P-1$ for 1D partitioning. Later, Boman et al. demonstrated that using hypergraph partitioning to determine the 2D structure provides an even better speedup in real-world graphs [7]. In previous work, we explored several possible data partitioning methods, evaluated on a high-performance supercomputer, in the context of reducing the runtime of the dimensionality reduction step of SPG [8]. While standard 1D partitioning allows the computation of the principal eigenvector of the residuals matrix of a 4-billion-vertex graph in approximately 5 minutes on 16384 cores, these 2D partitioning techniques enable much more favorable scaling properties by limiting the number of messages during inter-process communication. Fig. 1 illustrates the benefit observed in [8] of 2D partitioning. The figure shows the run time to find the first eigenvalue of a 2^{23}-vertex R-MAT matrix ([9]) as we increase the number of cores (strong scaling). In this plot, the performance difficulty of solving this eigensystem when utilizing a simple 1D random distribution (blue curve) is apparent. The scalability is clearly limited and the run time actually increases for more than 1024 cores. These 2D methods, however, show the improved scalability for two of these 2D distributions: one based on a random partitioning of rows/columns with an imposed 2D block Cartesian structure (2DR, red line, [6]) and one that uses hypergraph partitioning instead of random assignment of the rows/columns to improve the communication volume (2DH, green line, [7]). It is important to note that one nice property of the 2DH method is that the partitioning time is only slightly greater than the partitioning time of 1D hypergraph partitioning (2DH uses 1D hypergraph partitioning plus a small amount of additional work). Using these 2D methods, we are able to get more reasonable performance scaling and find eigenvectors for very large power-law graphs.

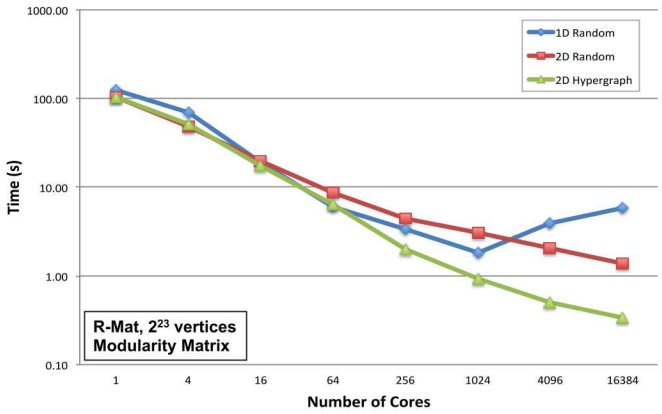

Fig. 1. Strong Scaling. Time (in seconds) to find one eigenvector when using different data distributions: 1D random, 2D Cartesian random, and 2D hypergraph random.

The 2DH method, in particular, scales well and is effective in reducing the runtime of the dimensionality reduction step. However, the challenge with the 2DH method is that it is relatively expensive to compute the distribution due to the expense of the hypergraph partitioning step, and it may not be possible to amortize this cost over enough SpMV operations. For instance, for an 8 million row R-MAT problem, approximately 40,000 SpMV operations are required to amortize the additional cost of calculating the 2DH distribution and yield a net improvement over the 2DR distribution (as shown in Fig. 2). Since we typically need at most a few thousand SpMV operations in our eigensolver, the partition must either (1) be useful for multiple time steps in a dynamic graph, or (2) be computable from a sampled graph which will take much less time, in order for the method to be computationally useful. In previous work, partitioning for dynamic graphs was explored and partitioning with sampling was introduced [8], in an effort to improve the intercept point in Fig. 2 and more easily amortize this computational expense.

In this paper, we focus on partitioning with sampling in order to enable the connectivity-based partitioning methods such as hypergraph partitioning more feasible for very large scale-free graphs. The remainder of this paper is organized as follows. In Section II, we describe the methodology used in our partitioning with sampling techniques and show some preliminary results. In Section III, we further explore sampling, applying several different sampling methods to improve the 2D hypergraph partitioning of three simple graph models, in an attempt to better understand why sampling and partitioning works. In Section IV, we explore the effect that graph structure has on partitioning and sampling. Section V provides a

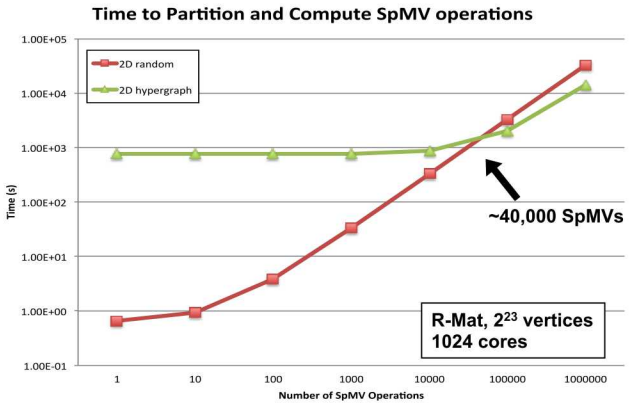

Fig. 2. Challenge with hypergraph partitioning. Time (in s) to partition once and compute multiple SpMV operations.

summary and discussion.

II. Partitioning and Sampling

Our previous study on the partitioning of dynamic graphs suggested to us that it might be possible to partition a simplified graph problem (one where edges have been removed) without significant loss of partitioning quality (i.e., without significant increase in SpMV execution time). Fig. 3 outlines our partitioning with sampling methodology. We first sample the edges of the input graph $G = (V, E)$ at rate α to obtain a new graph $G = (V, E')$, where $E' \subset E$ and $|E'| = \alpha|E|$. For the results in this section, we use simply uniform random sampling but in the subsequent section we explore more complicated sampling techniques. Next, we partition this significantly smaller graph G' and apply this partition to the original graph. The desired outcome is that G' is significantly less expensive to partition than G without a significant loss in the partitioning quality.

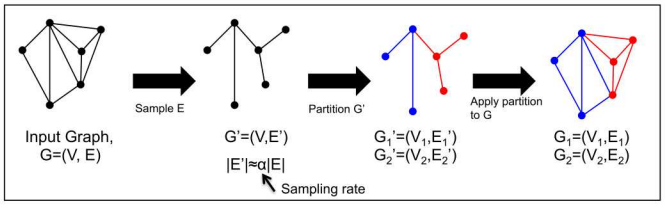

Fig. 3. Partitioning and sampling.

In preliminary results, this technique of partitioning a sampled graph has been very effective for several network graphs. Here and in subsequent results, we quantify partitioning quality as the time to compute a SpMV operation for a partitioned matrix. This SpMV operation was performed with the Epetra library, part of the Trilinos project [10]. All results here and in subsequent sections were generated on the NERSC Hopper supercomputer.

We found that—in a real, observed network— partitioning the graph based on a sample does not severely impact the running time of the SpMV. Using the hollywood movie actor network graph, hollywood-2009 [11], we see a great decrease in partitioning time when partitioning graphs that are formed by uniformly sampling the edges of the original graph (Fig. 4) with no significant increase in execution time of the resulting SpMV operation (Fig. 5) when these partitions are applied to the original graph. Revisiting the tradeoff between partitioning time and partitioning quality (SpMV runtime), we see that sampling can greatly reduce the number of SpMV operations needed to amortize the high partitioning time of the 2D hypergraph methods. For the hollwood-2009 graph, we see that applying a sampling rate of 0.1 can effectively reduce the number of SpMV operations needed to amortize the hypergraph partitioning methods by a factor of ten, making the 2DH method much more attractive. In this paper we investigate sampling techniques with respect to different graph features, to determine where this is an effective technique.

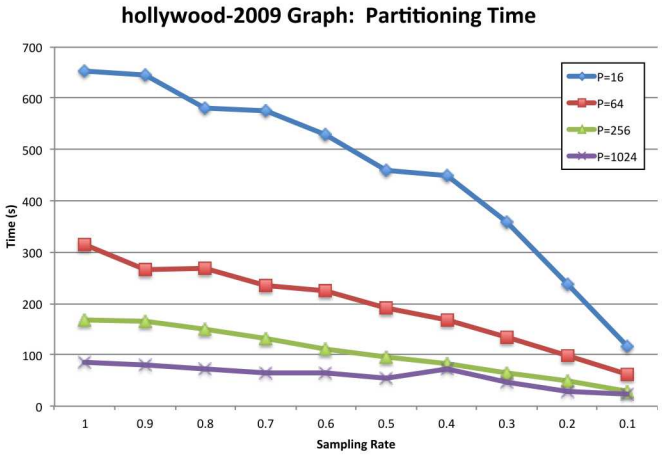

Fig. 4. Partitioning of sampled graph. Partitioning time in seconds for hollywood-2009 graph with the edges of the graph being sampled uniformly at different rates.

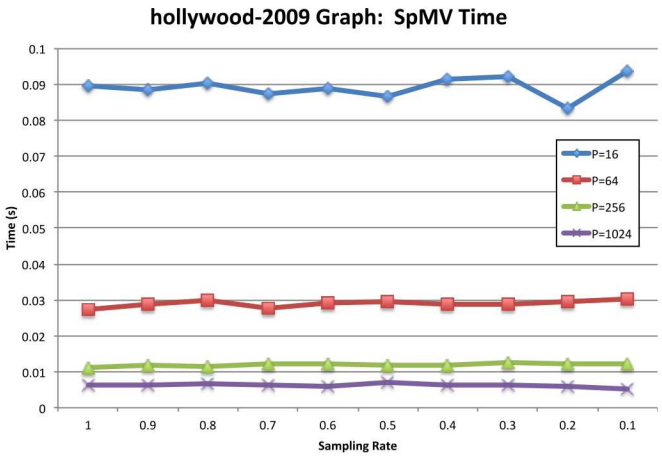

Fig. 5. Partitioning of sampled graph. SpMV time in seconds for hollywood-2009 graph when partitions of sampled graphs (with various sample rates) applied to original graph.

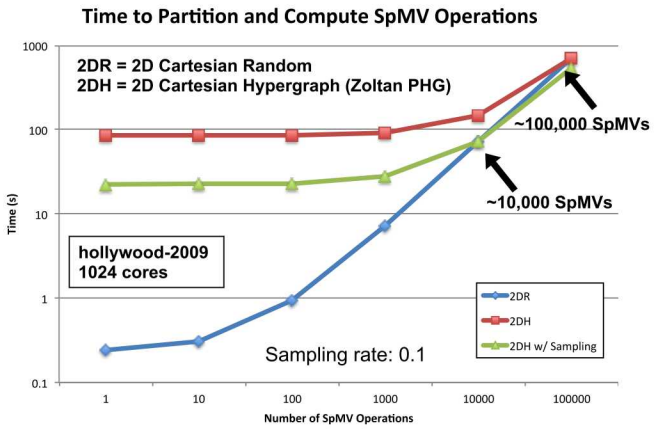

Fig. 6. Challenge with hypergraph partitioning (revisited).

III. Different Sampling Strategies for Network Graphs

In this section, we explore three different sampling techniques and their application to three different simple graph models. The first method is to sample graph edges in a uniformly random manner. The second method is to sample a graph edge with a probability proportional to degrees of the vertices contained by the edge (edges incident to high degree vertices are more likely to be chosen for the sampled graph). The final method is to sample a graph edge with a probability inversely proportional to degrees of its vertices (edges incident to high degree vertices are less likely to be chosen for the sampled graph).

We applied these three different sampling techniques to three simple graph models: the Erdős–Rényi model (where the edges are randomly chosen), the Chung-Lu model (where edges are randomly chosen to fit a specified degree distribution), and the stochastic block model (which approximates community structure in blocks by randomly choosing edges that connect two vertices within a block at a higher probability that edges that connect vertices in different blocks). The graph instances of each of these three models were built to have 2^{20} vertices and an average vertex degree of 32. For the Chung-Lu model, we sampled a beta distribution (a=2, b=5) in order to obtain the vertex degrees. A weight for each vertex is drawn from the beta distribution, and the probability of an edge between two vertices is defined by the product of their weights. Thus, the expected degree of a vertex is proportional to its weight (and, in fact, is equal to its weight times the sum of the weights over all vertices). The stochastic blockmodel was created with 32 communities of equal size, and probabilities within and between communities were set such that a vertex has an expected degree of 30 within its community and 2 outside of its community. The subsequent figures show the time to compute a 2DH partitioning with a particular sampling method for several different sampling rates and the resulting SpMV time relative to a 2DR partitioning.

Figure 7 shows the results of the three different sampling techniques for the Erdős–Rényi graph. As expected, sampling more sparsely greatly decreases the partitioning time (a decrease of up to 19 times). However, the hypergraph partitioning method does not improve the partitioning quality relative to the random method and produces significantly worse partitioning for most sampling rates. Since there is no inherently good partition for an Erdős–Rényi graph, any cut determined by the 2DH method will be based on pure happenstance, and sampling may cause good partitions to be missed.

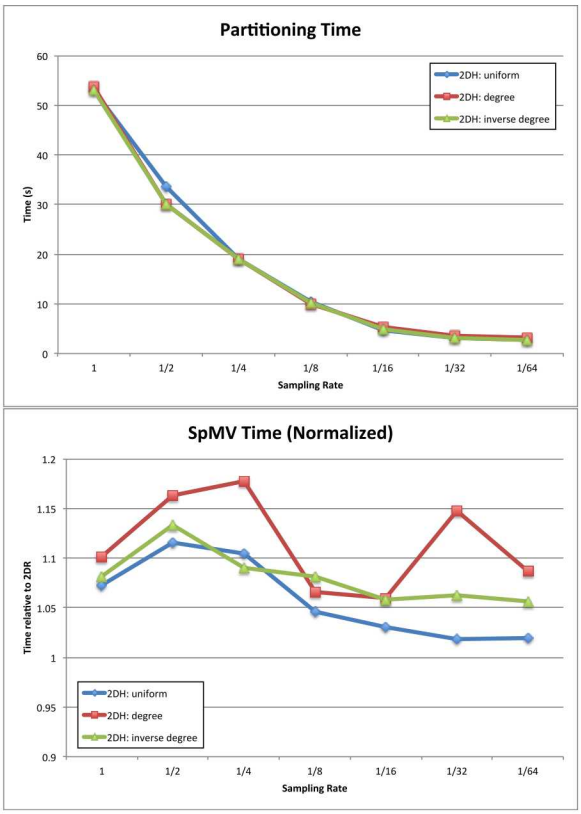

Fig. 7. Erdős–Rényi model. Time to partition the sparse matrix using 2DH with sampling (top) and time to perform the resulting SpMV operation relative to the resulting 2DR SpMV operation (bottom).

In Figure 8, we see the results of the three different sampling techniques when used to partition the Chung-Lu graph. Again, we see a significant (up to a factor of 18) decrease in partitioning time for the 2DH as we sample the edges less frequently (lower α). In terms of partitioning quality, the 2DH method with sampling produces mixed results. Uniform random and inverse degree sampling with 2DH partitioning produces partitions of worse quality and similar quality to 2DR, respectively. However, sampling proportionally to the vertex degrees produces partitions with 2DH significantly better than 2DR. Here, the skewed degree distribution creates some structure that can still be exploited within the sample.

In Figure 9, we see the results of the three different sampling techniques when used to partition the stochastic block model graph. Again, we see a significant (up to a factor of 11) decrease in partitioning time for the 2DH as we sample the edges less frequently (lower α). In terms of

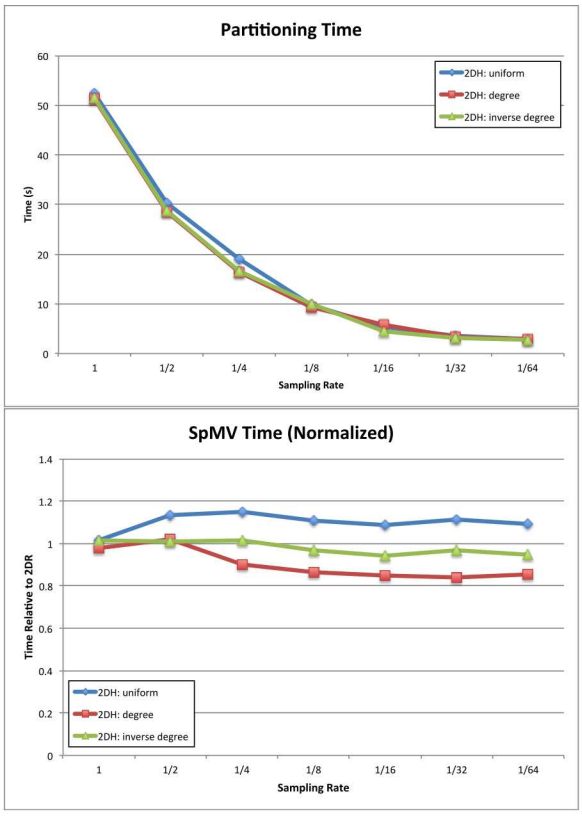

Fig. 8. Chung-Lu model. Time to partition the sparse matrix using 2DH with sampling (top) and time to perform the resulting SpMV operation relative to the resulting 2DR SpMV operation (bottom).

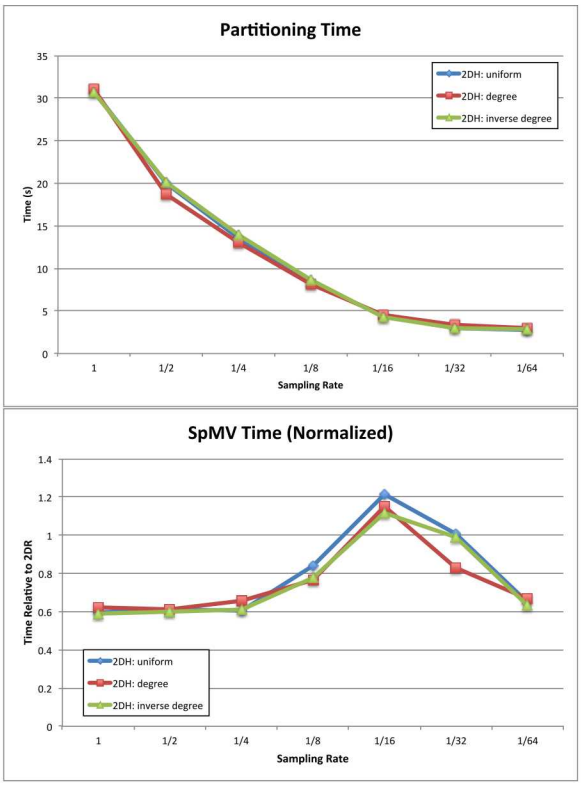

Fig. 9. Stochastic Block Model. Time to partition the sparse matrix using 2DH with sampling (top) and time to perform the resulting SpMV operation relative to the resulting 2DR SpMV operation (bottom).

partitioning quality, 2DH partitioning seems to be better for most of the sampling rates with all methods. Sampling does not significantly alter the partition quality at the sampling rates from $\alpha = 1/2$ to $\alpha = 1/8$. However, the quality does decrease significantly for $\alpha = 1/16$ and $\alpha = 1/32$ (except for the proportional degree sampling) before surprisingly improving again at $\alpha = 64$. In general, the sampling methods yield similar quality partitions with the degree and inverse-degree based methods being slightly better than uniform random.

IV. Effects of Graph Structure on Partitioning and Sampling

In this section, we explore the effect of community structure and degree distribution on partitioning and sampling in an attempt to further understand how this structure affects both the partitioning time and the partitioning quality [12].

We use R-MAT graphs with different parameters to simulate community structure in a graph. In particular, we generated R-MAT graphs with 2^{20} vertices and an average degree of 100 with the parameters a, $b = c = (0.75 - a)/2$, $d = 0.25$, where a was set to 0.25, 0.375, 0.5, and 0.625. Setting $a = 0.25$, the result will be an Erdős–Rényi graph, where all edges occur with equal probability (similar but more dense than the graph in the previous section). As a increases, the condition number of the probability matrix decreases, and the community structure in the graph increases as the two halves interact less. The matrices were partitioned with the random 2D method (2DR) and the hypergraph method (2DH). For the hypergraph method, we sampled $\alpha = 1/2^i$ of the edges at random for $0 \leq i \leq 10$.

Results are shown in Fig. 10. As expected, the partitioning time of the 2DH method is significantly reduced in all cases as the proportion of edges sampled decreases. The 2DH method reduces the SpMV runtime over the 2DR method in most cases for the values of a corresponding to more community structure (0.5 and 0.625), with at most a slight increase in runtime for $a < 0.5$. When the matrix is more similar to an Erdős–Rényi graph (like in Fig. 7), 2DH partitioning never significantly helps. This makes intuitive sense: Since there is no structure to the graph, partitioning randomly should provide about the same performance as looking for ideal cuts. One surprising aspect of the results is that, for the cases with more community structure, performance actually *improves* when the data were sampled. This may be an artifact of the R-MAT generator. It could be that, when there are fewer edges, the weak community structure of the R-MAT graph is more apparent, and as more edges between the two halves (and their recursive counterparts) are added, the heuristic used for partitioning method becomes less effective. This would imply that 2DH provides an advantage for SpMV, that is better revealed in the sparsified graph.

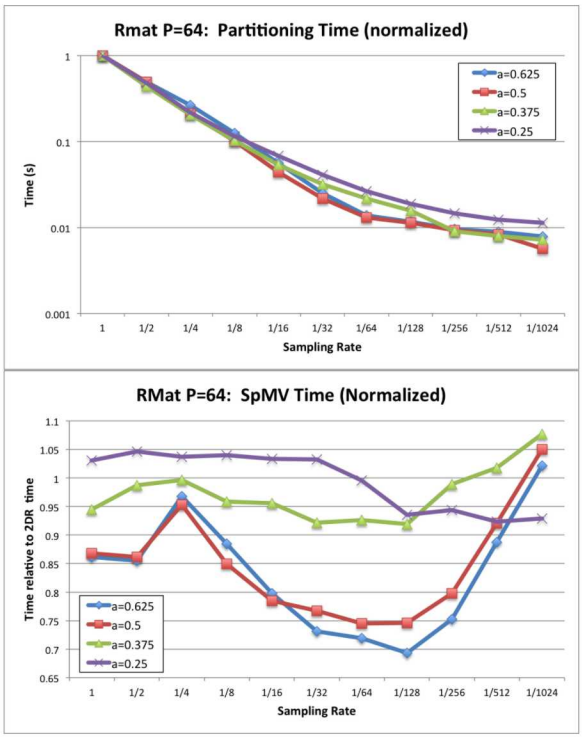

Fig. 10. Relative time to partition the sparse matrix (top) and perform a SpMV operation (bottom).

V. Summary/Conclusions

A key kernel in graph analytics is to solve very large eigensystems as part of dimensionality reduction. In order to solve these eigensystems efficiently, a good partitioning of the matrix is necessary. 2D distributions have been shown to be superior to their 1D counterparts (bounding the number of communication messages), with the 2DH method producing particularly high quality partitions. Unfortunately, this 2D hypergraph partitioning method is significantly more expensive than the random partitioning methods and this additional cost can be hard to amortize when solving an eigensystem resulting from a given graph. However, we have shown that is often possible to sample the graph, producing a reduced order approximate graph that can can be partitioned significantly faster with little reduction in partition quality.

In this paper, we explored three different random sampling techniques: uniform, degree-based, and inverse degree-based sampling. We applied these techniques to three simple graph models: Erdős–Rényi, Chung-Lu, and stochastic block model. In general, the degree-based and inverse degree-based methods performed better than the uniform method, with the exception of the random Erdős–Rényi graph. The 2D hypergraph method with sampling outperformed the 2D random method for the Chung-Lu and stochastic block model, but not for the random Erdős–Rényi graph. This make sense since the Chung-Lu and stochastic block models have some sparsity structure that the hypergraph based method may be able to use in order to improve upon a random structure. The significant improvement in running time with the stochastic blockmodel also suggests that community structure is the dominating driver of difficulty when partitioning to perform distributed SpMV operations.

We also used R-MAT matrices to explore to effect that community-like structure might have on partitioning and sampling, altering the R-MAT parameters to obtain a spectrum of the community-like structure. Again, with these experiments, we saw that those graphs with more structure led to significant better quality with the hypergraph partitioning in comparison to the random partitioning. For these graphs with significant community structure, sampling did not greatly reduce the partitioning quality until we sampled very sparsely (partitioning 1/256 of the edges).

In future work, experiments will also use the Block Two-level Erős–Rényi (BTER) generator [13], which more accurately models community structure. The R-MAT generator was used here for the sake of continuity with earlier work and since there is no publicly available parallel BTER generator.

Acknowledgment

The empirical results in this paper were computed using resources of the National Energy Research Scientific Computing Center, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

References

[1] B. A. Miller, N. T. Bliss, P. J. Wolfe, and M. S. Beard, "Detection theory for graphs," *Lincoln Laboratory J.*, vol. 20, no. 1, pp. 10–30, 2013.

[2] S. Fortunato, "Community detection in graphs," *Physics Reports*, vol. 486, pp. 75–174, February 2010.

[3] M. E. J. Newman, "Finding community structure in networks using the eigenvectors of matrices," *Phys. Rev. E*, vol. 74, no. 3, 2006.

[4] L. Page, S. Brin, R. Motwani, and T. Winograd, "The PageRank citation ranking: Bringing order to the web," Stanford InfoLab, Tech. Rep., 1999.

[5] U. V. Çatalyürek, C. Aykanat, and B. Uçar, "On two-dimensional sparse matrix partitioning: Models, methods, and a recipe," *SIAM J. Sci. Comput.*, vol. 32, no. 2, pp. 656–683, Feb. 2010. [Online]. Available: http://dx.doi.org/10.1137/080737770

[6] A. Yoo, A. H. Baker, R. Pearce, and V. E. Henson, "A scalable eigensolver for large scale-free graphs using 2D graph partitioning," in *Proc. Supercomputing*, 2011, pp. 63:1–63:11. [Online]. Available: http://doi.acm.org/10.1145/2063384.2063469

[7] E. G. Boman, K. D. Devine, and S. Rajamanickam, "Scalable matrix computations on large scale-free graphs using 2D graph partitioning," in *Proc. Supercomputing*, 2013, pp. 50:1–50:12. [Online]. Available: http://doi.acm.org/10.1145/2503210.2503293

[8] M. M. Wolf and B. A. Miller, "Sparse matrix partitioning for parallel eigenanalysis of large static and dynamic graphs," in *Proc. IEEE High Performance Extreme Comp. Conf.*, 2014.

[9] D. Chakrabarti, Y. Zhan, and C. Faloutsos, "R-MAT: A recursive model for graph mining," in *Proc. of the 4th SIAM Conference on Data Mining*, 2004, pp. 442–446.

[10] M. A. Heroux et al., "An overview of the Trilinos project," *ACM Trans. Math. Softw.*, vol. 31, no. 3, pp. 397–423, 2005.

[11] T. A. Davis and Y. Hu, "The University of Florida sparse matrix collection," *ACM Trans. Math. Softw.*, vol. 38, no. 1, pp. 1:1–1:25, Dec. 2011.

[12] M. M. Wolf and B. A. Miller, "Effects of graph structure on 2D partitioning of scale-free graphs with sampling," *SIAM Workshop Network Sci.*, 2015, to appear.

[13] C. Seshadhri et al., "Community structure and scale-free collections of Erdős-Rényi graphs," *Phys. Rev. E*, vol. 85, no. 5, p. 056109, 2012.