# Incremental Streaming Graph Partitioning

Lisa Durbeck
*Virginia Tech*
*Dept. of ECE*
ldurbeck @ vt.edu

Peter Athanas
*Virginia Tech*
*Dept. of ECE*
athanas @ vt.edu

***Abstract*—Graph partitioning is an NP-hard problem whose efficient approximation has long been a subject of interest. The I/O bounds of contemporary computing environments favor incremental or streaming graph partitioning methods. Methods have sought a balance between latency, simplicity, accuracy, and memory size. In this paper, we apply an incremental approach to streaming partitioning that tracks changes with a lightweight proxy to trigger partitioning as the clustering error increases. We evaluate its performance on the DARPA/MIT Graph Challenge streaming stochastic block partition dataset, and find that it can dramatically reduce the invocation of partitioning, which can provide an order of magnitude speedup.**

***Index Terms*—Computation (stat.CO); Graph partition; graph sparsification; spectral graph theory; LOBPCG; spectral graph partitioning; stochastic blockmodels; incremental partitioning; community detection**

## I. Introduction

Graphs are a species of big data that is particularly suited for representing relationships between entities. For example, a graph of a social network describes friendships and kinship. A graph partition is a more compact encoding of the data, in which the community structure of the graph is preserved, with a several-fold reduction in edges. Graph partition is among NP-hard problems for which exact solutions are too expensive to obtain because no polynomial-time solution is known; instead, people have developed heuristics to find clusters that approximate the optimal solution sufficiently well.

Today's phenomenal data production rates make it desirable to have a simple, fast and accurate graph partitioner along with convenient tools for graph analysis. The DARPA/MIT Graph Challenge (DGC) was envisioned as a way to accelerate progress in graph structure assessments such as graph isomorphism and graph partition [9]. One of its contributions is a set of graphs along with vertex group labels with which researchers can develop new algorithms and validate results. This also facilitates comparisons among competing approaches in terms of their computational and memory cost versus their accuracy of results. Facilitating this further is a graph generator, and a baseline algorithm and scoring functions against which researchers can benchmark their progress, and a code base in C++ or Python with which to begin. Within the larger DGC, which contains several challenges since 2017, this work focuses on the Streaming Graph Challenge (SGC), a challenge for graph partitioning that includes both static and streaming datasets of synthetic stochastic blockmodel-based (SBM) graphs.

This paper reports on efforts to advance progress further within the SGC by augmenting the fastest approach demonstrated to date by incorporating recent and complementary progress in *incremental* partition for time-varying graphs by Charisopoulos [2]. Partitioning is an expensive operation; we seek to reduce calls to the partitioner without impacting the partition quality. The approach takes advantage of intrinsic properties of the graph to track graph edge insertions and deletions, and indicate when the set of updates may cause a change to the community structure of the graph. When it does not, we retain the group labels from the prior partition—conserving partition cycles while tracking the optimal group assignments within an error tolerance $\epsilon$.

To better pinpoint the source of speedup we derive an expression for the rate of consumption of edge updates by the enhanced partitioning pipeline and show that under certain conditions it can produce an impressive speedup over the work/time of the partitioner, especially as the number of batches of updates increases. We experimentally verify these estimates for SGC SBM graphs with known partition labels.

The paper is organized as follows. The Background section outlines related work in graph partitioning motivating the approach used here, as described in the Approach and Methods section. The Experiments and Results section describes specifics on the datasets and computing environment, and the performance of the approach relative to the baseline and to the most relevant related work. Discussion and Future Work summarizes the work and suggests how it could be extended.

## II. Background

Many common development frameworks use the most rudimentary of the lightweight streaming partitioning techniques pioneered by Stanton and Kliot, still, such as hashing on node index [16]. Yet there are many potential benefits to approaches that trade some simplicity for greater computational or storage efficiency. A summary of approaches can be found in Fortunato [7].

Global approaches to graph partition rely on properties of the entire graph. The most common examples are spectral partitioning, which derives a partition from approximate eigenvectors of the adjacency matrix [8], and spectral clustering, which derives it using the eigen-decomposition of the graph Laplacian matrix [1], [14]. These were first developed in the 1960s and 70s, and took their modern form in the 1990s. Their major drawback is that the necessary access to information about the full graph may not be possible in practice for very large graphs—a drawback addressed to some degree in the current work by the use of a matrix-free method by Knyazev that does not require the explicit storage of the adjacency matrix [10].

The DGC baseline for streaming graph partition for stochastic block models (SBM) is a sequential implementation of a parallelizable Bayesian statistics-based stochastic block partitioning method of Peixoto [9], [13]. The baseline partitioner is accurate in its group assignments, but slow: its time-order is $\mathcal{O}(E \cdot log^2 E)$, where $E$ is the number of edges in the graph.

One of the winners in the 2017 SGC competition is a partitioner based on Locally Optimal Block Preconditioned Conjugate Gradient Method, or LOBPCG by Zhuzhunashvili and Knyazev [17]. LOBPCG is a spectral method particularly suited for sparse matrices and sparse graphs like those of the DGC stochastic block challenges [17]. LOBPCG has a faster time-order than the baseline of $\mathcal{O}(N \cdot k)$, where $N$ is the number of nodes in the graph, and $k$ is the number of groups, or clusters.

We confirmed these earlier-reported time-orders for both the Peixoto and Knyazev methods in the course of our experiments with SGC static graphs of sizes ranging from 50 nodes to 5 million nodes that were released for the challenge in 2017 or 2020. Since LOBPCG's time-order is a significant improvement over the baseline and gives similar accuracies, at least for the low-block-size, low-variation graphs within SGC, we based our experiments on seeing whether these ideas improve the performance of LOBPCG further.

LOBPCG is an efficient spectral clustering method. Other methods of spectral partitioning may be desirable in other situations. Spielman and Teng present a single-pass, local partitioning algorithm also based on spectral clustering using the eigenvalues of the graph Lapacian [15]. Their method uses $\mathcal{O}(N + E)$ storage and $\mathcal{O}(E)$ time where $E$ is the number of edges of the graph, and $N$ the number of nodes. LOBPCG has a smaller time-order and space-order due to the advantage of being a matrix-free method that does not require storing the graph coefficient matrix explicitly; instead, it can access the matrix by evaluating matrix-vector products [10].

To our knowledge, incremental partitioning was first introduced by Ou and Ranka, within the context of adaptive finite element meshing [12]. They observed that, for a class of problems such as adaptive finite element meshing, where continuous updates to the graph are occurring, the number of updates to the partition assignment at any given time is small relative to the size of the graph. They called this the incremental graph partitioning problem. They assigned partitions to graph updates using a simple local graph-distance-based metric.

A similar observation motivates Charisopoulos within the context of evolution of time-varying SBM graphs [2]. They developed a proxy for the Davis–Kahan bound on the subspace distance within updates to a graph adjacency matrix that serves to characterize graph updates succinctly in terms of distance from a spectrum threshold, within an incremental approach to spectral estimation, using subspace iteration for block Krylov methods. This provides a bound on the error for the accuracy of the subspace used to make the partition assignments.

We use a relevant distance metric for SBMs they derived and proved that requires access to degree information from the prior partition, and degree information within the current batch of graph updates [2].

Our partitioning algorithm is depicted in Figure 1 as a state machine. It functions similarly to Algorithm 3.1 in Charisopoulos [2]. The system has two states and three transitions; first, an accumulation state in which the low-cost distance metric corresponding with Equation 6 is periodically checked, and second, transition to a partitioning state in which the eigenvalues and eigenvectors are recalculated, after which the system returns to the accumulation state until the proxied-subspace distance is further than a distance of $\epsilon$ from the subspace generated in the prior partition cycle. In this way the partition label assignments remain within an error distance of $\epsilon$ from their true current values. This has the important feature of suppressing updates that do not substantially improve the subspace distance.

The distance metric we used is one formulated and demonstrated by Charisopoulos for gradually-evolving SBMs [2]. It provides a lightweight distance calculation for approximating the subspace update, and one more appropriate to sparse graphs than the usual 2-norm or spectral norm of the graph update matrix. Calculating this distance permits us to determine whether a batch of one or more updates has gotten too far away from the prior computed eigenvalues without actually recomputing the eigenvalues, which is far costlier in computation and memory usage than computing $d$ repeatedly.

We note that although the original equations depend minimally on aspects of the graph spectrum, the proxy does not depend upon anything that is not an intrinsic property of the graph easily obtained within any system that has access to the graph, regardless of whether it uses a spectral partitioning method. In particular, their use of the leading eigenspace leads to a maximum eigenvalue of 1 true

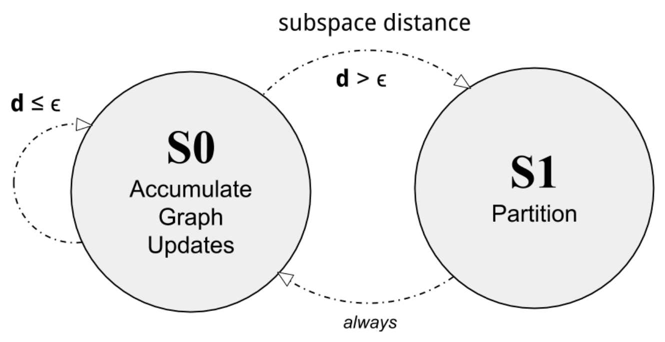

Fig. 1. State machine representation of algorithm as a process. The algorithm from a data view is encapsulated in Equation 5.

for all symmetric adjacency matrices [11]. We use this fact to set $d = 1$ and solve Equation 7 for $\alpha$.

## III. Approaches & Methods

To define the partitioning problem in general, given a graph with no group labels associated with any node, the goal is to produce accurate group label assignments for each node in the graph. An additional assumption for the present work is that the problem is an *incremental partition problem*. In such a case, the graph is revealed to the algorithm over time; it does not possess all the information about graph nodes and edges at the outset, and must produce the best group labels it can with the information it has at any specific time $t$, generally associate with the arrival of a new *batch* of graph nodes and/or edges.

To more formally define the incremental partition problem, we present the following terms. Let $G$ represent a graph as follows:

$$G = (N, E) \tag{1}$$

where $N$ is the set of nodes or vertices and $n$ is a node; $E$ is the set of edges, $e$ is an individual edge of the graph, and $e_i$ are the edges associated with node $i$. Let $P$ be the set of partitions of $G$ that we are seeking to generate by the method, of which there are a total of $k$, or $P = (p_1, p_2, \ldots, p_k)$. Each $p_i$ is a subset of $G$, and each vertex appears in at most one $p_i$.

The class of solvers this work focuses on is what has been termed *stateful streaming* by Kao [9]. That is, given a graph $G$, and the goal of evaluating a partitioning function $f(G)$, at the addition of an additional smaller graph $g$, the goal is to evaluate the function

$$f(G+g) \tag{2}$$

where $G$ is the already-partitioned portion of the graph, and $g$ is the new input to the partitioner. The intention is that $f$ uses information about $G$ in its assessment of $g$, unlike *stateless streaming*, in which $f(g)$ occurs without consideration of prior partition results from $G$.

More specifically, given a local partition function $h$ and a global partition function $j$, the proposed method evaluates the composition

$$P = j(h(G+d_i)) \tag{3}$$

where $d_i$ is the latest token in the input stream. The tokens in the test cases used here are typically a batch of edge insertions that may introduce new nodes to the graph, but more generally they can be $g$, a new subgraph added to $G$; or they can be a set of node- or edge insertions or deletions, in the case of a temporally-varying graph.

The function $j()$ simply returns $h(G+d_i)$ unless a global repartitioning criterion is met. The repartitioning criterion can be as simple as

$$|G| \mod c \tag{4}$$

where $c$ is heuristically derived; or, the criterion could be based on an independent assessment of the quality deterioration of $P$, run in tandem or in parallel with the streaming partition, as it is in the present work. In the present work, Equation 3 is equivalent to

$$P = S_1(S_0(G+d_i)) \tag{5}$$

That is, the local partitioner $h()$ is simply an accumulator and does not re-partition; $h$ is achieved by the $S_0$ node of Figure 1 and its actions, and $j()$ is achieved by the $S_1$ node, i.e. partition of $G$ along with $d_i$, the set of insertions and deletions to the adjacency matrix of $G$. As in incremental partitioning, the global partitioner of the function $j$ serves as the more-accurate partitioner that is invoked to limit the accumulation of error associated with local-information-based updates to $P$.

The partition assignment $P$ is continually produced by a constant-time streaming partitioner that is periodically rebalanced by the results of a highly accurate linear-time global spectral method used for $j()$, with much more limited information used in the interim by $h()$.

For the repartitioning criterion, this work investigates the utility of the distance proxy of Charisopoulos [2] that was designed with SBMs in mind, envisoning the evolution of a graph via incoming graph updates. They derive a simple bound on the accuracy of the computed block Krylov subspace of a graph matrix under streaming edge updates that bounds the error for the accuracy of the leading subspace used to make partition assignments. Proposition 4.1 of their derivation and its corollaries provide the following distance criterion for the set of changes to the graph since the last partition at time $t$:

$$\frac{1}{\rho(A_t)} \geq \epsilon \geq d \tag{6}$$

where $\rho(A_t)$ is the spectral radius of the graph corresponding with the maximum eigenvalue,

$$d = \alpha + \kappa\alpha + \frac{\alpha}{(1+\alpha)^2} \tag{7}$$

for which $\kappa$ is based on the current largest and smallest node degrees in the most recent graph partition at time $t$, in magnitude, irrespective of edge direction,

$$\kappa = \sqrt{\frac{deg_{max}(G)}{deg_{min}(G)}} \tag{8}$$

and $\alpha$ is similarly based on the largest change in node degree from the current set of graph updates. Given the edge insertions and deletions in $g$, $\alpha$ is set once per batch, or $d_i$, to the maximum ratio within the graph of edge insertions and deletions to any node relative to its former state at time $t$

$$\alpha = max\left(\frac{|e_i \in g|}{|e_i \in G|}\right) \tag{9}$$

over all $i \in g$. Note that $|e_i \in G|$ is no different from the degree of node $i$ at time $t$ but written here to correspond with the changes in $d_i$. Charisopoulos discusses and demonstrates the order-of-magnitude

TABLE I
MACHINE PROPERTIES FOR EXPERIMENTATION

| | |
|---|---|
| Memory size | 512 GB |
| Cores | 64 |
| Processor | AMD Opteron 6376 |
| Clock Frequency | 2.3 GHz |
| Floating point | 64-bit double precision |
| Python | 3.6.9 |

increase in triggers to partition that this results in over a change criterion based on the 2 norm of the changes to the adjacency matrix, i.e. $A_{t+1} - A_t$; however, they argue that this stricter partitioning criterion tracks changes better in sparse graphs. This is the formulation of the change criterion used here, and the distance $d$ in Figure 1. Since the leading eigenvalue of the normalized Laplacian is $1$ for symmetric adjacency matrices, the triggering criterion reduces to testing whether $d < 1$.

The global function this work investigates is locally-optimal block partition conjugate gradient, a spectral clustering method developed by Knyazev in 2001 [10] that has been demonstrated as a significant improvement to the traditionally used Lanczos method. Zhuzhunashvili and Knyazev demonstrated the utility of this spectral clustering method within the context of the SGC [17]. They reported very high partition quality, with 99% or better accuracy against the known truth partition, for both static and streaming graph partition with the Challenge graphs, for the 2017 SGC dataset of graphs of size 50,000 nodes to 2M nodes, and found a 100-1000$\times$ speedup over the baseline partitioner. These are the best speedup results to date in this public challenge, by an order of magnitude or more. They observed the performance within the experimental set of static graphs to be $\mathcal{O}(n \cdot k)$.

Local optimality implies that LOBPCG will converge on a solution at least as fast as the gradient descent method, and this has been both theoretically guaranteed and practically observed [10]. For the SGC datasets, Zhuzhunashvili and Knyazev found convergence occurred three times faster at later stages in the stream processing, for which the partial results from earlier stages provided a better initial seed than the random one used in initial stages [17]. Our implementation employs this seeding, or *warm start* as well.

This work uses the DARPA-MIT Stochastic Streaming Graph Challenge metrics to assess the methods described in Kao [9], in the convenient form implemented within the SGC baseline support functions. These metrics include pairwise precision and recall, and overall accuracy, defined as the percentage of nodes correctly partitioned, which is the percent match of the algorithm's results with the true labels. In all cases the evaluation first finds the mapping between the algorithm's group labels and those of the known truth data.

In addition we evaluate our algorithm in terms of its speedup over the baseline, and over our implementation of Zhuzhunashvili and Knyazev's LOBPCG-based approach with and without our modification based on Charisopoulos' repartition criteria. Speedup is typically defined in terms of latency as $t/W$ of the new algorithm over $t/W$ of the old, where $t$ is the total execution time and $W$ is the workload. Here we report $t$ in number of cycles, where a cycle is defined as a call to the partitioner and labeler. Times are reported for the entire execution time, including loading and transforming inputs, and total execution time is $sys + user$, all time spent on the user's behalf either within the code or within the kernel reported by the Unix utility `time`.

The synthetic SBM graphs that are integral to SGC have known partition label assignments, or *ground truth*. The dataset contains multiple sets of graphs differing in size but having similar characteristics to one another, such as probability of inter- and intra-cluster edges, maximum and minimum node degrees, as well as groups with similar block size, between-block interactions, and block size variation [4], [9]. They range in size from 500 to 5M vertices. We worked with both the original set of graphs from 2017 and those released in 2020.

## IV. Experiments

The accuracy and time-complexity of the proposed approach was evaluated using graphs with known partition solutions. Typically, three to ten trials were performed for each experiment; the results represent the mean. Charisopoulos' approach maintains the accuracy of the subspace used to make partition assignments within a small tolerance $\epsilon$ of the previous partition results as edge updates

TABLE II
MEAN ALGORITHM PERFORMANCE MEASUREMENTS, FOR GRAPHS GROUPED BY GRAPH SIZE, FOR THE BASELINE P (PEIXOTO) ALGORITHM VERSUS L (ENHANCED-LOBPCG) ALGORITHM.

| GRAPH $\frac{\lvert N\rvert}{\lvert E\rvert}$ | ALG | $\frac{5K}{100K}$ | $\frac{20K}{400K}$ | $\frac{50K}{1M}$ | $\frac{200K}{5M}$ | $\frac{1M}{24M}$ | $\frac{2M}{41M}$ | $\frac{5M}{230M}$ |
|---|---|---|---|---|---|---|---|---|
| TIME (s) | P | 1,373 | 16,868 | 80,893 | | | | |
| | L′ | 196 | 463 | 898 | 3,658 | 28,642 | 107,591 | 404,986 |
| SPEEDUP | L′ | 7.0 | 36.4 | 90.1 | | | | |
| ACCURACY | P | 0.77 | 0.85 | 0.80 | | | | |
| | L′ | 0.99 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| PAIRWISE | P | 0.86 | 0.89 | 0.68 | | | | |
| PRECISION | L′ | 0.99 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| PAIRWISE | P | 0.78 | 0.86 | 0.93 | | | | |
| RECALL | L′ | 0.99 | 1.00 | 1.00 | 1.00 | 1.00 | 0.99 | 1.00 |

come in. We took a batched approach to inputs, but varied batch size from the given $|E|/10$ in the `emergingEdges` dataset downward to arbitrary sizes as low as two graph updates per batch. After each batch, we use our simple proxy for right-sizing the next batchsize as the process moves through the input stream, and we trigger repartition when either that amount of input has been consumed, or when an individual node's accumulated changes triggers a repartition.

Similar to Zhuzhunashvili and Knyazev, we use the clusterQR method by Damle [3] to assign labels based on the eigenvectors. One important difference is the formulation of the graph Laplacian used. Since Charisopoulos' formulation is based upon the largest eigenvalues, we follow their assumptions, including the use of the normalized Laplacian $L = D^{-1/2}WD^{-1/2}$ in a search for the largest eigenvalues. This had similar ultimate accuracy to what Zhuzhunashvili and Knyazev reported, which held for the more challenging 2020 graphs. It has the added benefit of converging to a near-optimal solution faster than they reported for the nonnormalized Laplacian $L = D - W$, typically reaching its solution at around Stage 3, or 30% of the graph edge data.

Experiments were carried out using Python implementations of the algorithms on a server-class machine with 512 GB of memory. The machine characteristics are summarized in Table I. The memory size allowed us to run LOBPCG against the 2 million- and 5 million-node graphs without using its block handling capabilities. This machine was unable to run the baseline algorithm in Python to completion on graphs larger than 50,000 nodes; however, similar limits were reported earlier by other participants in SGC.

TABLE III
MEAN REPARTITIONING CYCLES FOR THE LOBPCG ALGORITHM BEFORE AND AFTER AUGMENTATION. SPEEDUP, PREDICTED SPEEDUP, AND THEIR RELATIVE DIFFERENCE.

| LOBPCG CYCLES | LOBPCG′ CYCLES | SPEEDUP | PREDICTED | REL. DIFF |
|---|---|---|---|---|
| $10^1$ | 20 | 0.49 | 0.78 | 0.37 |
| $10^2$ | 30 | 3.37 | 3.92 | 0.14 |
| $10^3$ | 36 | 27.52 | 26.12 | 0.05 |
| $10^4$ | 43 | 234.38 | 195.89 | 0.20 |
| $10^5$ | 46 | 2,153.85 | 1,567.15 | 0.37 |
| $10^6$ | 50 | 19,867.55 | 13,059.55 | 0.52 |

While we took our starting point from the `emergingEdges` dataset, we generated the input stream for our experiments as a random permutation of the static representation of the graph. This better fit Charisopoulos' observations about the update edgeset of an evolving time-varying SBM graph. If updates are randomly chosen from the distribution of edges in the graph, then the edgelist set in Charisopoulos' formulation extends naturally to all nodes in $G$.

For something such as filling in a static graph from an input stream via a random permutation of edges, one would expect the graph structure to change with the log of the changes to the graph size, as proxied by $\kappa$ and $\alpha$. This is maintained at the node level as well, albeit with a simpler criterion for $\alpha$. A sequence of random changes to the graph is unlikely to change the graph structure so long as the number of changes increases the number of edges by $\alpha$, where $\alpha$ is as defined in Equation 7.

This, in turn, suggests that using the Charisopoulos conditional where appropriate results in maintenance of partition quality with fewer and fewer repartitions as the graph grows, with the total number of repartitions given by $log_{1+\alpha}(|E|/e_0)$, where $e_0$ is the initial input batch size. We refer the reader to an upcoming publication of this work for a more detailed look at this property [6].

We demonstrate this speedup and its dependence on batch size by constructing a longer input stream out of the given SGC graphs by reducing the batch size by one, two, three and more orders of magnitude, reducing the initial batch size $e_0$. Table III shows the results. While initially at batch size of $|E|/10$ this approach actually is slower-performance, at the next decrement, each row is an order of magnitude increase in the number of batches into which the input is divided, and each produces an order of magnitude Speedup. The fit of the results to the above log of the changes is shown as Predicted Speedup and the Relative Difference; the correspondence is fairly good.

## V. Discussion & Future Work

Providing a lightweight, change-based partition triggering criterion enhances the performance of existing partitioning algorithms by choking down the partition rate to as-necessary rather than as-preordained. We show how effective a log-proxy-based choke mechanism is at keeping the number of calls to the partitioner low without sacrificing accuracy. This provides a useful rationale and potential mechanism for accumulating updates until they pass a certain error threshold in the label assignments. Having etablished that this *can* have benefits, we now are working to better understand its limits, from an analysis of its sensitivity to temporal ordering within the SGC stream datasets such as snowball sampling.

Our code implementing the Charisopoulos repartitioning criterion in Python is available on GitHub [5]. Next, we intend to assess the time and memory performance at a finer grain, and investigate further performance gains for graph partition from hardware acceleration.

## VI. Acknowledgements

The authors would like to thank Andrei Knyazev and Nicholas Macias for helpful discussions and advice; and the anonymous reviewers for helpful comments and suggestions. This research was supported in part by an appointment to the Intelligence Community Postdoctoral Research Fellowship Program at the National Institute of Standards and Technology (NIST), administered by Oak Ridge Institute for Science and Education (ORISE) through an interagency agreement between the U.S. Department of Energy and the Office of the Director of National Intelligence (ODNI).

## References

[1] William N Anderson Jr and Thomas D Morley. Eigenvalues of the laplacian of a graph. 1971.

[2] Vasileios Charisopoulos, Austin R Benson, and Anil Damle. Incrementally updated spectral embeddings. *arXiv preprint arXiv:1909.01188*, 2019.

[3] Anil Damle, Victor Minden, and Lexing Ying. Robust and efficient multi-way spectral clustering. *arXiv preprint arXiv:1609.08251*, 2016.

[4] DARPA/MIT. Data sets | graphchallenge. *https://graphchallenge.mit.edu/data-sets*, 2016.

[5] L Durbeck. Charisopoulos partitioning criterion. *GitHub repository https://github.com/ldurbeck/charisopoulos*, 2020.

[6] L Durbeck and P Athanas. Incremental streaming graph partitioning (manuscript in preparation).

[7] Santo Fortunato. Community detection in graphs. *Physics reports*, 486(3-5):75–174, 2010.

[8] Alan J Hoffman. Some recent results on spectral properties of graphs. *Beiträge zur Graphentheorie*, pages 75–80, 1968.

[9] Edward Kao, Vijay Gadepally, Michael Hurley, Michael Jones, Jeremy Kepner, Sanjeev Mohindra, Paul Monticciolo, Albert Reuther, Siddharth Samsi, William Song, et al. Streaming graph challenge: Stochastic block partition. In *2017 IEEE High Performance Extreme Computing Conference (HPEC)*, pages 1–12. IEEE, 2017.

[10] Andrew V Knyazev. Toward the optimal preconditioned eigensolver: Locally optimal block preconditioned conjugate gradient method. *SIAM journal on scientific computing*, 23(2):517–541, 2001.

[11] Jiri Matoušek. On approximate geometric k-clustering. *Discrete & Computational Geometry*, 24(1):61–84, 2000.

[12] Chao-Wei Ou and Sanjay Ranka. Parallel incremental graph partitioning. *IEEE Transactions on Parallel and Distributed Systems*, 8(8):884–896, 1997.

[13] Tiago P Peixoto. Efficient monte carlo and greedy heuristic for the inference of stochastic block models. *Physical Review E*, 89(1):012804, 2014.

[14] Alex Pothen, Horst D Simon, and Kang-Pu Liou. Partitioning sparse matrices with eigenvectors of graphs. *SIAM journal on matrix analysis and applications*, 11(3):430–452, 1990.

[15] Daniel A Spielman and Shang-Hua Teng. A local clustering algorithm for massive graphs and its application to nearly linear time graph partitioning. *SIAM Journal on computing*, 42(1):1–26, 2013.

[16] Isabelle Stanton and Gabriel Kliot. Streaming graph partitioning for large distributed graphs. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1222–1230, 2012.

[17] David Zhuzhunashvili and Andrew Knyazev. Preconditioned spectral clustering for stochastic block partition streaming graph challenge (preliminary version at arxiv.). In *2017 IEEE High Performance Extreme Computing Conference (HPEC)*, pages 1–6. IEEE, 2017.