LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

LLNL-CONF-812693

Scaling Graph Clustering with Distributed Sketches

B. W. Priest , A. M. Dunton, G. Sanders

July 21, 2020

IEEE High Performance Extreme Computing - Graph Challenge
Waltham, MA, United States
September 22, 2020 through September 24, 2020

Disclaimer

Scaling Graph Clustering with Distributed Sketches

Benjamin W. Priest
Center for Applied Scientific Computing
Lawrence Livermore National Laboratory
Livermore, CA, USA
priest2@llnl.gov

Alec Dunton
Department of Applied Mathematics
University of Colorado Boulder
Boulder, CO, USA
alec.dunton@colorado.edu

Geoffrey Sanders
Center for Applied Scientific Computing
Lawrence Livermore National Laboratory
Livermore, CA, USA
sanders29@llnl.gov

***Abstract*—The unsupervised learning of community structure, in particular the partitioning vertices into clusters or communities, is a canonical and well-studied problem in exploratory graph analysis. However, like most graph analyses the introduction of immense scale presents challenges to traditional methods. Spectral clustering in distributed memory, for example, requires hundreds of expensive bulk-synchronous communication rounds to compute an embedding of vertices to a few eigenvectors of a graph associated matrix. Furthermore, the whole computation may need to be repeated if the underlying graph changes some low percentage of edge updates. We present a method inspired by spectral clustering where we instead use matrix sketches derived from random dimension-reducing projections. We show that our method produces embeddings that yield performant clustering results given a fully-dynamic stochastic block model stream using both the fast Johnson-Lindenstrauss and CountSketch transforms. We also discuss the effects of stochastic block model parameters upon the required dimensionality of the subsequent embeddings, and show how random projections could significantly improve the performance of graph clustering in distributed memory.**

***Index Terms*—streaming algorithms, random matrices, graph clustering**

I. Introduction

Analysts working in applications as widely ranging as biology, sociology, network science, computer science, telecommunications, and others deal regularly with graph-expressible data where a major task of interest is to find structure, usually defined as a partitioning of "like" vertices into clusters. Although many graph clustering algorithms have arisen in the literature, and several works make important contributions to improving their distributed versions [1]–[4], applying them to scales that require distributed implementations on modern systems (topology data is near PetaScale, say $>$ 100B edges) remains challenging.

We will center our focus in this paper to accelerating algorithms in the form of spectral clustering [5]. In the most generic form, spectral clustering computes approximate eigenpairs of a graph-dependent matrix (commonly adjacency, Laplacian, or a centered/normalized variant). Then, k extremal eigenvectors form an *embedding* of the vertices into $\mathbb{R}^k$. Some downstream conventional clustering algorithm (e.g. K-means [6] or dbscan [7]) then partitions the vertices using their embedded locations. Often recursion is employed, where the initial embedding reveals several of the *best* partitions (*strongest* communities) and reapplying the process on much smaller sub-partitions to further resolve community structure.

Although spectral clustering is a popular method and reasonably performant (log-linear cost and storage in input data) for real-world serial applications, it is not currently employed at extreme scales. State-of-the-art eigensolvers such as Krylov/Lanczos methods are iterative methods, relying on sparse linear algebraic kernels (primarily sparse matvec, dot-product, and their block vector variants), for which high-quality scientific computing packages are available [8], [9]. Recent work develops these linear algebra kernels and others for challenging graph topology [10]. Distributed memory implementations of eigensolvers require hundreds of resource-hungry bulk-synchronous operations (sequences of sparse matvecs). Numerical stopping criteria are also poorly understood for general, large real-world graphs. Finally, conventional spectral embedding algorithms are not efficient in evolving graph applications, where an update of a few edges (say those connecting previously poorly connected partitions) can cause previous solutions to take longer to iteratively improve than random initial guesses would.

In this document we present an alternative method relying upon a cheaper and more scalable embedding procedure using *linear sketches*: dimension-reducing linear projections drawn from a carefully-chosen distribution. Random sketch matrices allow us to approximately preserve row inner products and norms of an arbitrary matrix in a much lower dimension with high probability. Our method replaces an expensive spectral embedding with an efficient - though coarse - linear sketch embedding. In addition to gaining computational efficiency, linear sketches are designed for the turnstile streaming data model - i.e. they are indifferent to the order in which matrix items are received, and are robust to changes in the underlying matrix. Thus, our method produces a vertex embedding that features (1) computation and communication linear in the number of edge updates, (2) a simple distributed memory implementation, and (3) natural robustness to dynamic data.

Others have broadly applied matrix sketches throughout numerical linear algebra, often with an emphasis on matrix multiplication, regression, or low-rank approximation [11].

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344 (LLNL-CONF-812693) and was supported by the LLNL-LDRD Program under Project No. 20-FS-037. Experiments were performed at the Livermore Computing Facility.

Ailon and Chazelle introduced a fast formulation of the classic Johnson-Lindenstrauss transform (JLT, [12]) with the fast approximation of nearest neighbors in the embedding space as a motivating example [13]. Traganitis et al. devised a library of tools SkeVa that utilize sampling to produce approximate K-means clustering on high-dimensional data [14]. Our approach is partially inspired by Gilbert et al., who used JLTs to directly, though coarsely, approximate the singular values and vectors of large matrices [15].

Many recent graph and matrix sketching applications make use of linear ℓ_p-sampling sketches [16] based upon precision sampling [17]. Ahn et al. described spectral clustering algorithms on hypergraphs, and used sampling sketches to accelerate their algorithms [18]. Ahn, Guha and McGregor applied linear sampling sketches to estimate graph properties [19], [20] including approximating spectral sparsification [21], which Kapralov et al. improved with a single-pass algorithm [22]. Unfortunately, ℓ_p-sampling sketches are tricky to implement and are not considered efficient for large scale applications.

Others have utilized sketches as a means to accelerate spectral clustering. Fowlkes et al. [23] describe a method using a Nyström low-rank approximation to the Laplacian by way of column sampling, and later using the eigenspectrum of the Nyström approximation to construct an embedding to be clustered in $\mathcal{O}(nmk+m^3)$ operations, where n is the number of vertices, m is the dimension of the Nyström approximation, and k is the embedding dimension. Li et al. [24] later improved the complexity to $\mathcal{O}(nmk)$. We will describe a method that requires $\mathcal{O}(\mathrm{nnz}(X))$ time, where $\mathrm{nnz}(X)$ is the number of nonzero elements in the graph matrix X - i.e. twice the number of edges. Gittens et al. use a conventional power iteration method to coarsely approximate eigenvectors before utilizing JLTs to project into a small dimensional space [25]. Our method avoids the expensive power iteration process. Additionally, unlike either of these methods, our embeddings scale naturally to the distributed model, take advantage of the sparsity structure in the graph matrix, and are fully robust to dynamic and streaming graphs.

We consider popular efficient random matrix projections in our analyses: the fast Johnson-Lindenstrauss transform based upon the Walsh-Hadamard transform (FWHT) [13], and the CountSketch transform (CST) [26]. We will produce results comparing the performance of FWHT- and CST-based embeddings of stochastic block model (SBM) vertices into lower-dimension space. In our experiments, we employ UMAP [27] to perform an additional non-linear dimensionality reduction and HDBSCAN [28] to cluster the resulting embeddings. We further present scaling results demonstrating that our linear embedding procedure can embed SBMs with tens of billions of edges in seconds on a modest compute cluster.

II. Notations and Background

We will assume throughout an undirected, unweighted, unsigned, connected and static graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with vertex and edge sets of size $|\mathcal{V}| = n$ and $|\mathcal{E}| = m$, respectively. Spectral clustering methods generalize trivially to weighted graphs and other have extended them to handle signed graphs [29], dynamic graphs [30], and directed graphs [31]. The sketching methods we employ generalize to weighted, signed, dynamic, and directed graphs with much less complication, as the key operator is a simple linear projection. In cases where signal to noise ratio is high enough, the sketching approach likely provides scalability that is not possible in the existing approaches. For example, some directed graph methods involve complex-valued iterative eigensolvers, which are challenging to implement efficiently at scale on graph topologies. This is particularly the case for sketch-based dynamic graphs [32].

In particular, we will assume that $\mathcal{G}$ is sampled from an undirected stochastic block model with c communities (or blocks) with symmetric probability matrix $P \in [0,1]^{c\times c}$ and the community size vector $\mathcal{C} \in \mathbb{Z}_+^c$. The (i,j)th entry of P indicates the pairwise probability that each of the $\mathcal{C}_i$ vertices in community i are neighbors with each of the $\mathcal{C}_j$ vertices in community j. In general, the diagonal entries of P will be larger than the off-diagonal entries, which is meant to simulate the inter- and intra-connection densities of ground truth communities in empirical networks. The expected ratio between the diagonal entries of P and and off-diagonal entries of P are determined by additional parameters ρ_{in} and ρ_{out}, which parameterize the distributions from which the elements of P are sampled.

A graph $\mathcal{G}$ has adjacency matrix A, where $A_{x,y} = 1$ iff $xy \in \mathcal{E}$ and is zero otherwise. We will embed the rows of the adjacency matrix because the SBMs we consider are nearly regular, and so little is gained by utilizing a Laplacian formulation. We use *pythonic* notation for matrix rows and columns (the row vector corresponding to the neighborhood of vertex i is $A_{i,:}$ and the i-th column is $A_{:,i}$).

Canonical adjacency-based spectral embedding of dimension k on $\mathcal{G}$ is performed as follows:

1) Let $V \in \mathbb{R}^{n\times k}$ be the matrix whose columns $V_{:,1}, \ldots, V_{:,k}$ are the eigenvector of A associated with most positive eigenvalues.
2) Identify with each vertex x the row vector $V_{x,:}$.

We call the rows of V a *spectral embedding* of their corresponding vertices. The spectral clustering algorithm consists of computing a spectral embedding and executing a clustering algorithm on the embedded vectors, i.e. the rows of V.

III. Matrix Sketching

Matrix sketching is a numerical linear algebraic tool with applications in, e.g., latent semantic indexing [33], low-rank approximation [34], and least-squares problems [35]. A primary goal of matrix sketching is to embed a matrix $X \in \mathbb{R}^{n\times p}$, comprised of n data points in a p-dimensional feature space, in a lower dimensional space such that geometric properties of the original matrix are preserved to a desired level of fidelity. Mathematically, this typically entails the application of a linear

operator $S \in \mathbb{R}^{p \times s}$ with $s \ll p$ to form the *sketch matrix* $XS \in \mathbb{R}^{n \times s}$. [1]

For the scope of this work, we seek sketching matrices that

1) preserve pairwise distances between rows in X to within a tolerance which we denote ε, with a constant failure probability;
2) satisfy 1) by projecting into $\Theta(\varepsilon^{-2} \log n)$ dimensions;
3) admit scalable and sparse distributed memory implementations.

To this end we consider the FWHT, which satisfies conditions 1) and 2), but we will show that it struggles to satisfy our needs for condition 3). We also consider the CST and show that it exhibits superior distributed memory scaling as compared to FWHT, although it does not provide as strict a guarantee for condition 1). In particular, CST is *not* a Johnson-Lindenstrauss transform. However, it has been applied to great effect throughout the literature even with its less strict guarantee [11], [26], [36]. We extensively show that CST and FWHT create embeddings of similar quality in our experiments in Section IV.

In rough terms, the Johnson-Lindenstrauss lemma states that there is distribution of linear operators that can embed any n points in a p dimensional feature space into $\Theta(\varepsilon^{-2} \log(n))$ dimensions such that all pairwise inner products are preserved to within a factor of $(1 \pm \varepsilon)$ with a constant probability of failure [12]. That is, our embedding dimension is independent of the dimension of the feature space and logarithmically dependent on the number of points which we are embedding. The FWHT embeds a matrix X as $XDPS \in \mathbb{R}^{n \times s}$ where $D \in \mathbb{R}^{p \times p}$ is a diagonal matrix whose entries are i.i.d. ± 1 with equal probability, $P \in \mathbb{R}^{p \times p}$ is a Hadamard matrix, and $S \in \mathbb{R}^{p \times s}$ is a sparse matrix whose nonzero entries are one to indicate uniform subsampling. In practice p is assumed to be a power of two. This construction allows the FWHT to satisfy the Johnson-Lindenstrauss lemma with asymptotically lower complexity than prior JLT formulations.

D, P, and S are never actually formed in practice, as their entries can be generated quickly as needed. In particular, embedding an element $X_{i,j}$ amounts to sampling column indices $k_1, \ldots k_s$ and generating a vector $X_{i,j} * D_{i,i} * [H_{j,k_1}, \ldots H_{j,k_s}]$.

The CST was first developed for numerical linear algebra by Clarkson and Woodruff [26] and was inspired by the celebrated CountSketch [37]. The sketch is computed as XR, where $R \in \{-1, 0, 1\}^{p \times s}$ is a sparse matrix with 1 non-zero element per column that is ± 1 with equal probability. The nonzero column indices and nonzero values are computable using 2-universal hash functions, obviating the expensive i.i.d. sampling requirements of the FWHT.

Importantly, computing XR CST requires only $\mathcal{O}(\mathrm{nnz}(X))$ operations. Also important, the rows of XR preserve the sparsity of the rows of X. FWHT embeddings will always be dense - even if the corresponding row has only 1 nonzero element!

[1] Similar sketching procedures involving multiplication on the left-hand side of the argument, as well as those which embed both the row-space and column-space of X (bi-linear sketches), can be defined analogously.

These features make CST particularly suited to distributed online sketching of graphs, in which edges arrive one-by-one into working memory simultaneously on a large number of processors as highly sparse vector updates. However, the FWHT satisfies more stringent theoretical guarantees, making it a useful baseline for comparison.

It is import to note that both FWHT and CST can be implemented in a fully-dynamic streaming fashion on arbitrary matrices that arrive in any order and can evolve during the process of performing the embedding. This makes FWHT and CST embeddings fully robust to changes in the underlying graph, a large advantage over the practical difficulties that face spectral methods in the streaming setting.

IV. Clustering Experiments

A. Experimental Setup

In our experiments we implement an embedding-clustering pipeline similar to that of spectral clustering. In particular, we use a sketch transform to produce an embedding into some dimension $s = \Theta(\varepsilon^{-2} \log n)$, and then further sharpen the pairwise distances between communities by applying the nonlinear dimensionality reduction tool UMAP. We assume throughout that we know the ground truth number of communities c for each graph considered, and so we use UMAP to reduce the embedding dimension from s to c. We then cluster the rows of the embedding in $\mathbb{R}^{n \times c}$ using HDBSCAN. We keep the parameters of both UMAP or HDBSCAN fixed at their default values in all experiments.

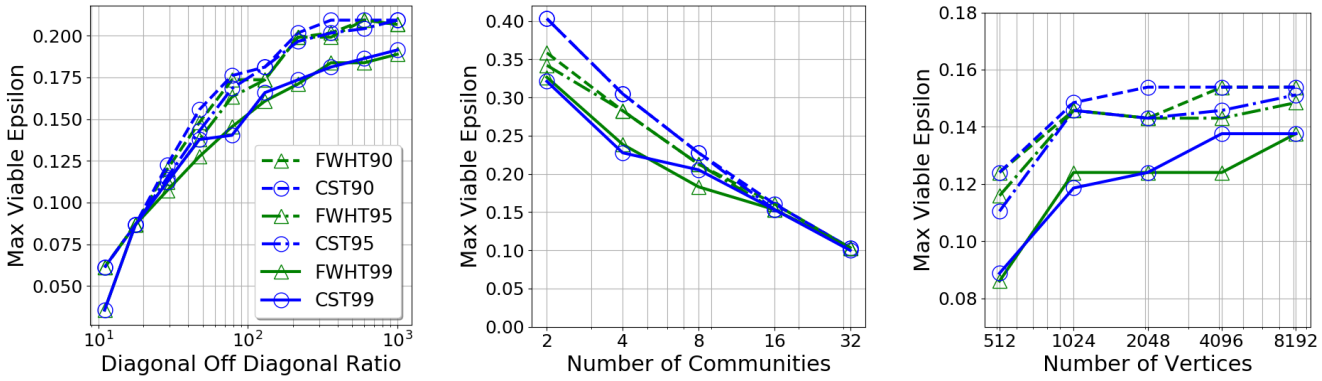
In our clustering experiments we assess the performance of our Sketch-UMAP-HDBSCAN pipeline using pairwise precision and recall [38] as our primary metrics. In particular, we determine the relationship between the parameter ε - which determines the fidelity to which inner products are preserved by our embedding - with the following SBM features:

1) the ratio of on-diagonal to off-diagonal entries in the probability matrix $\frac{\rho_{in}}{\rho_{out}}$;
2) the number of communities c;
3) the number of vertices n.

We test embeddings produced by both CST and FWHT, making the simplifying assumption that all SBMs have a probability matrix where diagonal elements equal ρ_{in} and off-diagonal elements equal ρ_{out}. We also assume that all blocks are equally-sized, i.e. that $\overline{\mathcal{C}} = n/c$. We apply our methods to more sophisticated SBMs that violate these assumptions in the next section.

In our experiments, we track what we refer to as the *maximum viable* ε. We define this to be the largest value ε can achieve in a sketch embedding before either the pairwise recall or pairwise precision falls below a certain threshold. This maximum viable ε is related to the sketch dimension of the graph matrix in an inverse squared manner [12]; e.g., a reduction of epsilon by a factor of 10 leads to a 100-fold increase in embedding dimension.

In the first experiment, we vary the on- versus off-diagonal ratio of ρ_{in} to ρ_{out} from $\theta(10)$ to $\theta(10^3)$, fixing the number



(a) On- vs Off-Diagonal Ratio (b) Community Count Scaling (c) Vertex Count Scaling

Fig. 1. The estimated maximum viable ε as a function of SBM parameters. Figure 1(a) plots ε as a function of the ratio of the on-diagonal to off-diagonal elements in the probability matrix associated with a SBM with 16 communities and 256 vertices per community. Figure 1(b) plots ε as a function of the number of communities in an SBM with 4096 vertices and an on-diagonal to off-diagonal p matrix entry ratio of ~ 50. Figure 1(c) plots ε as a function of the number of vertices in an SBM with 16 equally-sized communities and an on-diagonal to off-diagonal p matrix entry ratio of ~ 50.

of vertices $n = 4096$, the number of communities $c = 16$ and community sizes to $\mathcal{C}_i = 256$ for each i. The row sums of the generated P matrix are set to be 0.5, and for each sketch/parameter combination we run 10 independent trials. In the second two experiments we fix $\rho_{in}/\rho_{out} \sim 50$ and $\mathcal{C}_i = n/c$, varying c in experiment two and n in experiment three. In the second experiment, we fix $n = 4096$ while varying c from 2 to 32. In the third experiment, we fix $c = 16$ while varying n from 512 to 8096. In all three experiments we set the thresholds for precision and recall (the *metric threshold*) to 0.90, 0.95, and 0.99. Our metric threshold dictates how large we can allow ε to grow while maintaining the given tolerance.

B. Results

Examining the first test case with results reported in Figure 1(a), we see that as the on-diagonal to off-diagonal ratio is increased, the maximum viable ε increases. This matches our intuition; if connections between different communities are unlikely relative to those within communities, low-dimensional embeddings more accurately preserve clustering features. This corresponds directly to looser bounds on ε; if our clusters are more isolated from one another, the sketch embedding can be constructed such that it preserves geometric structure to a lesser extent. We also observe that the CST and FWHT perform nearly identically for all test cases, giving credibility to the use of the more distributed computationally-friendly CST in place of the more theoretically justified FWHT. Finally, as we decrease our metric thresholds from 0.99 to 0.90, we obtain a larger maximum viable ε; when we decrease our demands on cluster quality, our embedding can be of lower fidelity.

We now determine the dependence of the maximum viable ε on the number of communities in our SBM. We expect that, as we increase the number of communities with the number of vertices fixed at 4096, achieving desired precision/recall will become more difficult. Consequently, our sketch will have to map into a higher embedding dimension to sufficiently capture the geometry of the original matrix, corresponding to a lower maximum viable ε. Our results shown in Figure 1(b) confirm this prediction; as we increase the number of communities in our SBM we see that the maximum viable ε grows smaller.

Finally, we fix the number of communities in our SBM while increasing the overall number of vertices in the SBM (hence we are increasing the number of vertices per community as well). Figure 1(c) shows that the maximum viable ε increases as the number of vertices increases Further, as the number of vertices increases, we observe a flattening out of the maximum viable ε, which indicates that our methods ought to scale well as we increase the size of the graphs we are clustering, assuming that the community count remains fixed.

Across all three experiments we observe some important trends. First, we see that increasing our metric threshold does not drastically decrease the value of ε necessary to embed our graph matrix. We therefore expect high performance from our methods at a relatively marginal cost. Further, the CST and FWHT achieve quite similar results in all test cases. Given that the CST is much faster and naturally implemented in a distributed setting while the FWHT satisfies more rigorous embedding properties, this is a particularly exciting result. Broadly, our results suggest that we ought to expect upper bounds on ε to be increase as the number of communities increases, as the number of vertices per community decreases, and as the overlap between communities increases. On the other hand, as clusters in a graph become less disparate, our sketch embedding must capture the properties of the full graph matrix to a higher degree of fidelity.

V. Scaling Experiments

We now analyze the quality of the embedding-clustering pipeline on large benchmark graphs and test the scalability of our distributed-memory implementation of the embedding procedure. We utilize a selection of open-source SBMs generated as a part of the HPEC graph challenge [38]. Unlike our earlier experiments, these SBMs feature variable-sized

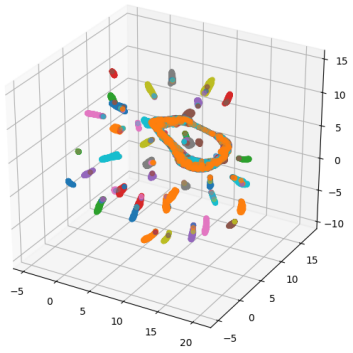


Fig. 2. A 3-dimensional visualization of the clusters produced by projecting a 50 thousand vertex SBM with 44 communities using CST with $\varepsilon = 0.01$. Vertex embedding locations are colored according to their ground truth partition. Although the corresponding row of Table I shows poor clustering results, a simple visual inspection shows that many of the true partitions separated cleanly, even in only 3 dimensions. The discrepancy in analytic metrics can likely be ameliorated by recursion and/or exploration of the UMAP/HDBSCAN parameter space, which we have chosen to keep fixed.

$\lvert\mathcal{V}\rvert$	**Pair Precision, Recall** > 0.9 **Parameters**				
	# Partitions	ε	***PP***	***PR***	***Accuracy***
500	8	0.1	0.96983	0.97719	0.986
1000	11	0.1	0.95991	0.95301	0.976
5000	19	0.05	0.97103	0.97395	0.9878
20000	32	0.018	0.91305	0.90455	0.9588
50000	44	0.01	0.55959	0.12414	0.73773

TABLE I
SBM EMBEDDING EXPERIMENTS

communities and more complex probability matrices. Table I shows a selection of graph sizes, true cluster counts, and the values of ε used to produce a CST embedding, as well as the pair precision (PP), pair recall (PR), and accuracy averaged over 10 independent trials.[2] In general, ε was chosen to obtain average PP and PR both > 0.9. We note, however, than as the graphs grow in size and complexity, that ε also decreases. Figure 1(b) provides a likely explanation for this phenomenon, as community count serves as a damping factor on the maximum viable ε. Further, performant UMAP and HDBSCAN parameters most likely differ as SBMs vary in size and complexity.

Figure 2 shows the largest of these SBM embeddings projected down into 3 dimensions and colored according to their ground truth communities. As we can see, even in the small dimensional space, the embedding manages to separate most of the clusters, some completely, others mixed into clusters of 2 or more communities. This suggests that our method, like most others at scale, will likely be best applied by hierarchically partitioning and refining subsets of vertices.

A. Distributed Sketching

A sketch embedding of dimension s on a square graph matrix $X \in \mathbb{R}^{n\times n}$ of $\mathcal{G}$ is performed as follows:

1) Choose a desired precision $\varepsilon \in (0, 1)$.
2) Sample a sketch operator $S \in \mathbb{R}^{n\times s}$, with $s = O(\varepsilon^{-2})$.
3) Compute sketch XS.
4) Identify with each vertex x the row vector $(XS)_{x,:}$.

We distribute this procedure as follows. Assume a universe of processors $\mathcal{P}$, and further assume some arbitrary balanced partitioning of vertices to processors $f : \mathcal{V} \to \mathcal{P}$.[3] We will abuse notation and refer to the set of vertices assigned to $P \in \mathcal{P}$ by f as $f^{-1}(P)$. Let $\boldsymbol{\sigma}$ be an arbitrary stream of edge updates defining $\mathcal{E}$, partitioned such that each $P \in \mathcal{P}$ receives the substream $\boldsymbol{\sigma}_P$. Each $P \in \mathcal{P}$ maintains a sketch vector in $\mathbb{R}^s$ corresponding to $(XS)_{x,:}$ for each $x \in f^{-1}(P)$. On reading an edge $uv \in \boldsymbol{\sigma}_P$, P sends uv to $f(u)$ and vu to $f(v)$. Upon receiving an edge xy, $f(x) = P$, processor P updates $(XS)_{x,:}$ appropriately. After having read over $\boldsymbol{\sigma}$ and cleared their communication buffers, $\mathcal{P}$ has XS stored in distributed memory.

We examine the scaling limits of the embedding procedure by implementing our distributed sketches using the C++/MPI communication library YGM [39] and applying them to very large SBMs generated with GraphChallenge 2017 parameters, which we fit with constrained regression. We use

$$c = 0.95 * n^{-0.36}, \overline{\mathcal{C}} = .95n^{0.64}, Var(\mathcal{C}) = .32n^{0.64},$$
$$\rho_{in} \sim 16.75n^{-0.59}, \rho_{out} \sim -1.02n^{-0.59},$$

where c is the number of communities, $\overline{\mathcal{C}}, Var(\mathcal{C})$ are the parameters used for sampling community sizes, and ρ_{in}, ρ_{out} are internal/external edge density parameters used in the SBM generation. All of our distributed experiments were performed on a cluster of Intel Xeon E5-2695 processors each featuring 36 cores.

Figure 3(a) shows the wall time scaling of our codes with a fixed number of processors as graph size increases. Figure 3(b) shows the wall time where instead the graph to be embedded is fixed and we increase the number of compute nodes. Finally, Figure 3(c) shows the scaling where only the embedding dimension increases - i.e. ε decreases. These scaling studies reinforce our assertion of the fitness of CST for generating high quality low-dimensional embeddings for clustering applications, and highlight the weaknesses of the more disciplined but also more cumbersome FWHT. In particular, we find that our implementation scales at a rate no worse than reading the graph into memory.

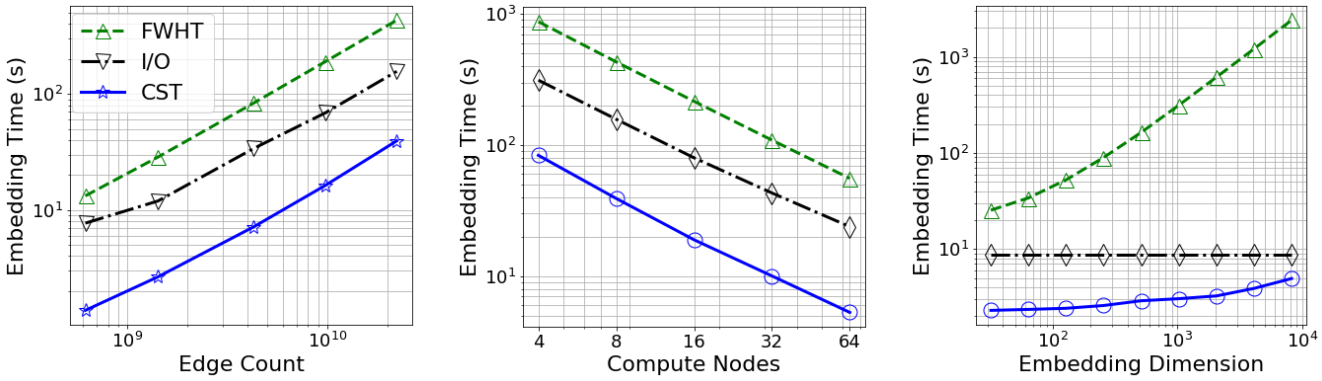
VI. CONCLUSIONS AND FUTURE WORK

We have demonstrated a scalable vertex embedding procedure using linear sketches, and have validated its utility and scalability on partitioning SBMs. We have shown that this approach provides an algorithmic workflow similar to spectral clustering at a fraction of the cost, and with the benefit of much higher scalability.

A major limiting factor to our analysis at scale is that we have limited our scope to local-parallelism-only implementations of clustering algorithms. However, we have shown

[2]Results using FWHT were similar.

[3]We use simple round-robin assignment in our experiments.



(a) Fixed Resource Scaling (b) Strong Scaling (c) Embedding Dimension Scaling

Fig. 3. The distributed memory scaling of applying CST and FWHT to SBMs of various sizes. We also plot the total time spent reading the graphs from file for comparison. Figure 3(a) plots the wall time for a set of 4 compute nodes as the edge count of the graph to be embedded into 128 dimensions increases from $\sim$ 500 million to $\sim$ 22 billion. Figure 3(b) plots the wall time for embedding a fixed-size graph with 200 million vertices ($\sim$ 22 billion edges) into 128 dimensions as the number of compute nodes increases. Both Figures 3(a) and 3(b) feature roughly linear scaling, as desired. Figure 3(c) plots the wall time for four compute nodes to embed a 20 million vertex graph as the embedding dimension increases, i.e. as ε decreases.

that our embedding algorithm features excellent scalability in distributed memory, and is able to embed graphs with tens of billions of edges in seconds on modest hardware. Further, our sensitivity experiments suggest that for a fixed desired precision, we find embeddings that yield clusters whose quality has at most a small dependence upon n, although the dependance upon the number of true communities and their size variance warrants further investigation. In future work we will demonstrate the scalability of the full clustering pipeline to distributed memory data scales by the introduction of novel distributed clustering algorithms.

It is further important to recall that, though SBMs feature convenient analytical properties, they do not reflect many properties of real graphs found in applications. Indeed, one of the largest challenges associated with distributed graph algorithms is managing the communication and computation bottlenecks introduced by the presence of high-degree vertices in scale-free graphs. Degree-Corrected Stochastic Block Models (DCSBMs) generalize SBMs with power-law degree distributions so as to more accurately simulate this feature. Detection of large dense regions (degree-corrected quasi-cliques) injected into real-world graphs would also be an important validation step. We will augment our algorithms in future work to manage high-degree vertices via *vertex delegation* [40] and sparse sketch storage.

Vertex embedding has many applications within graph machine learning beyond clustering. For example, recent vertex representation learning efforts such as node2vec [41] utilize deep neural networks to construct an embedding of vertices into low-dimensional latent space. Scaling to massive graph sizes, however, remains challenging. We believe that linear sketch-based embedding such as what we have proposed could significantly scale such nonlinear embeddings at a negligible cost to the representation quality.

ACKNOWLEDGEMENTS

The authors would like to thank Van Emden Henson for helpful comments and discussion.

REFERENCES

[1] X. Que, F. Checconi, F. Petrini, and J. A. Gunnels, "Scalable community detection with the Louvain algorithm," in *2015 IEEE International Parallel and Distributed Processing Symposium*, 2015, pp. 28–37.

[2] M. Halappanavar, H. Lu, A. Kalyanaraman, and A. Tumeo, "Scalable static and dynamic community detection using grappolo," in *2017 IEEE High Performance Extreme Computing Conference (HPEC)*, 2017, pp. 1–6.

[3] S. Ghosh, M. Halappanavar, A. Tumeo, and A. Kalyanarainan, "Scaling and quality of modularity optimization methods for graph clustering," in *2019 IEEE High Performance Extreme Computing Conference (HPEC)*, 2019, pp. 1–6.

[4] X. Liu, J. S. Firoz, M. Zalewski, M. Halappanavar, K. J. Barker, A. Lumsdaine, and A. H. Gebremedhin, "Distributed direction-optimizing label propagation for community detection," in *2019 IEEE High Performance Extreme Computing Conference (HPEC)*, 2019, pp. 1–6.

[5] U. Von Luxburg, "A tutorial on spectral clustering," *Statistics and computing*, vol. 17, no. 4, pp. 395–416, 2007.

[6] S. Lloyd, "Least squares quantization in PCM," *IEEE transactions on information theory*, vol. 28, no. 2, pp. 129–137, 1982.

[7] M. Hahsler, M. Piekenbrock, and D. Doran, "dbscan: Fast density-based clustering with r," *Journal of Statistical Software*, vol. 91, no. 1, pp. 1–30, 2019.

[8] V. Hernandez, J. E. Roman, and V. Vidal, "SLEPc: A scalable and flexible toolkit for the solution of eigenvalue problems," *ACM Trans. Math. Software*, vol. 31, no. 3, pp. 351–362, 2005.

[9] C. G. Baker, U. L. Hetmaniuk, R. B. Lehoucq, and H. K. Thornquist, "Anasazi software for the numerical solution of large-scale eigenvalue problems," *ACM Trans. Math. Softw.*, vol. 36, no. 3, Jul. 2009. [Online]. Available: https://doi.org/10.1145/1527286.1527287

[10] A. Buluç and J. R. Gilbert, "The combinatorial BLAS: Design, implementation, and applications," *The International Journal of High Performance Computing Applications*, vol. 25, no. 4, pp. 496–509, 2011.

[11] D. P. Woodruff *et al.*, "Sketching as a tool for numerical linear algebra," *Foundations and Trends® in Theoretical Computer Science*, vol. 10, no. 1–2, pp. 1–157, 2014.

[12] W. B. Johnson and J. Lindenstrauss, "Extensions of Lipschitz mappings into a Hilbert space," *Contemporary mathematics*, vol. 26, no. 189-206, p. 1, 1984.

[13] N. Ailon and B. Chazelle, "The fast Johnson–Lindenstrauss transform and approximate nearest neighbors," *SIAM Journal on computing*, vol. 39, no. 1, pp. 302–322, 2009.

[14] P. A. Traganitis, K. Slavakis, and G. B. Giannakis, "Sketch and validate for big data clustering," *IEEE Journal of Selected Topics in Signal Processing*, vol. 9, no. 4, pp. 678–690, 2015.

[15] A. C. Gilbert, J. Y. Park, and M. B. Wakin, "Sketched SVD: Recovering spectral features from compressive measurements," *arXiv preprint arXiv:1211.0361*, 2012.

[16] H. Jowhari, M. Sağlam, and G. Tardos, "Tight bounds for lp samplers, finding duplicates in streams, and related problems," in *Proceedings of the thirtieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. ACM, 2011, pp. 49–58.

[17] A. Andoni, R. Krauthgamer, and K. Onak, "Streaming algorithms via precision sampling," in *Foundations of Computer Science (FOCS), 2011 IEEE 52nd Annual Symposium on*. IEEE, 2011, pp. 363–372.

[18] K. Ahn, K. Lee, and C. Suh, "Hypergraph spectral clustering in the weighted stochastic block model," *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 5, pp. 959–974, 2018.

[19] K. J. Ahn, S. Guha, and A. McGregor, "Analyzing graph structure via linear measurements," in *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms*. SIAM, 2012, pp. 459–467.

[20] ——, "Graph sketches: sparsification, spanners, and subgraphs," in *Proceedings of the 31st ACM SIGMOD-SIGACT-SIGAI symposium on Principles of Database Systems*. ACM, 2012, pp. 5–14.

[21] ——, "Spectral sparsification in dynamic graph streams," in *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*. Springer, 2013, pp. 1–10.

[22] M. Kapralov, Y. T. Lee, C. Musco, C. Musco, and A. Sidford, "Single pass spectral sparsification in dynamic streams," *SIAM Journal on Computing*, vol. 46, no. 1, pp. 456–477, 2017.

[23] C. Fowlkes, S. Belongie, F. Chung, and J. Malik, "Spectral grouping using the Nyström method," *IEEE transactions on pattern analysis and machine intelligence*, vol. 26, no. 2, pp. 214–225, 2004.

[24] M. Li, X.-C. Lian, J. T. Kwok, and B.-L. Lu, "Time and space efficient spectral clustering via column sampling," in *CVPR 2011*. IEEE, 2011, pp. 2297–2304.

[25] A. Gittens, P. Kambadur, and C. Boutsidis, "Approximate spectral clustering via randomized sketching," *Ebay/IBM Research Technical Report*, 2013.

[26] K. L. Clarkson and D. P. Woodruff, "Low-rank approximation and regression in input sparsity time," *Journal of the ACM (JACM)*, vol. 63, no. 6, pp. 1–45, 2017.

[27] L. McInnes, J. Healy, and J. Melville, "UMAP: Uniform manifold approximation and projection for dimension reduction," *arXiv preprint arXiv:1802.03426*, 2018.

[28] L. McInnes, J. Healy, and S. Astels, "hdbscan: Hierarchical density based clustering," *Journal of Open Source Software*, vol. 2, no. 11, p. 205, 2017.

[29] A. Knyazev, "On spectral partitioning of signed graphs," in *2018 Proceedings of the Seventh SIAM Workshop on Combinatorial Scientific Computing*. SIAM, 2018, pp. 11–22.

[30] H. Ning, W. Xu, Y. Chi, Y. Gong, and T. S. Huang, "Incremental spectral clustering with application to monitoring of evolving blog communities," in *Proceedings of the Seventh SIAM International Conference on Data Mining, April 26-28, 2007, Minneapolis, Minnesota, USA*. SIAM, 2007, pp. 261–272. [Online]. Available: https://doi.org/10.1137/1.9781611972771.24

[31] H. Van Lierde, T. W. S. Chow, and J.-C. Delvenne, "Spectral clustering algorithms for the detection of clusters in block-cyclic and block-acyclic graphs," *Journal of Complex Networks*, vol. 7, no. 1, pp. 1–53, 05 2018. [Online]. Available: https://doi.org/10.1093/comnet/cny011

[32] L. Martin, A. Loukas, and P. Vandergheynst, "Fast approximate spectral clustering for dynamic networks," 2017.

[33] C. H. Papadimitriou, P. Raghavan, H. Tamaki, and S. Vempala, "Latent semantic indexing: A probabilistic analysis," *Journal of Computer and System Sciences*, vol. 61, no. 2, pp. 217–235, 2000.

[34] N. Halko, P.-G. Martinsson, and J. A. Tropp, "Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions," *SIAM review*, vol. 53, no. 2, pp. 217–288, 2011.

[35] V. Rokhlin and M. Tygert, "A fast randomized algorithm for overdetermined linear least-squares regression," *PNAS*, vol. 105, no. 36, pp. 13 212–13 217, 2008.

[36] F. Yang, S. Liu, E. Dobriban, and D. P. Woodruff, "How to reduce dimension with PCA and random projections?" *arXiv preprint arXiv:2005.00511*, 2020.

[37] M. Charikar, K. Chen, and M. Farach-Colton, "Finding frequent items in data streams," *Theoretical Computer Science*, vol. 312, no. 1, pp. 3–15, 2004.

[38] E. Kao, V. Gadepally, M. Hurley, M. Jones, J. Kepner, S. Mohindra, P. Monticciolo, A. Reuther, S. Samsi, W. Song *et al.*, "Streaming graph challenge: Stochastic block partition," in *2017 IEEE High Performance Extreme Computing Conference (HPEC)*. IEEE, 2017, pp. 1–12.

[39] B. Priest, T. Steil, R. Pearce, and G. Sanders, "You've Got Mail: Building missing asynchronous communication primitives," in *Proceedings of the 2019 International Conference on Supercomputing*. ACM, 2019, p. 8.

[40] R. Pearce, M. Gokhale, and N. M. Amato, "Faster parallel traversal of scale free graphs at extreme scale with vertex delegates," in *High Performance Computing, Networking, Storage and Analysis, SC14: International Conference for*. IEEE, 2014, pp. 549–559.

[41] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 2016, pp. 855–864.