Advantages to Modeling Relational Data using Hypergraphs versus Graphs

Michael M. Wolf, Alicia M. Klinvex, and Daniel M. Dunlavy
Center for Computing Research, Sandia National Laboratories
Albuquerque, NM 87185
{mmwolf,amklinv,dmdunla}@sandia.gov

***Abstract*—Driven by the importance of relational aspects of data to decision-making, graph algorithms have been developed, based on simplified pairwise relationships, to solve a variety of problems. However, evidence has shown that hypergraphs—generalizations of graphs with (hyper)edges that connect any number of vertices—can better model complex, non-pairwise relationships in data and lead to better informed decisions. In this work, we compare graph and hypergraph models in the context of spectral clustering. For these problems, we demonstrate that hypergraphs are computationally more efficient and can better model complex, non-pairwise relationships for many datasets.**

I. Introduction/Background

A. Relational Data and Graph Models

Many applications in the data sciences require the analysis of very large datasets (e.g., billions of objects), identifying important groups, structure, and trends in the data. Data analysts are often tasked with making timely, informed decisions based on the analysis of these very large and often noisy data sets. Relational data, which describes the relationships between objects, is commonly used in these analyses.

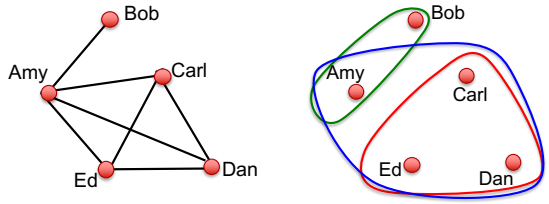
Driven by the importance of relational aspects of data to high-consequence decisions (e.g., decisions where it is crucial to minimize false positives and/or false negatives), there has been much effort in developing algorithms and software based on graph representations of data. A graph is a mathematical representation of a set of objects (vertices) and the pairwise connections (edges) between those objects (vertices). Graphs are common representations of relational data, where complex, non-pairwise relationships are translated to multiple pairwise relationships (edges) as illustrated concretely with the email example in Figs. 1 and 2a. Fig. 1 shows a summary table of an email dataset, where each "x" designates that a specific user was involved in a specific email. Fig. 2a shows the corresponding graph representation of this email data, where each of the users is represented by a vertex in the graph. Two graph vertices are connected with an edge if and only if the corresponding users participated in an email together.

This simple graph model, however, suffers from several shortcomings. One difficulty is that information is lost when the complex, non-pairwise relationships are represented by a collection of pairwise edges. For instance, from the graph in Fig. 2a, it is unclear that Carl, Dan, and Ed participated in the same email (as they did in email 2). This same graph could also represent that these three users participated

Emails

Users	1	2	3
Amy	x		x
Bob	x		
Carl		x	x
Dan		x	x
Ed		x	x

Fig. 1. Summary table for email data, which summarizes which of the five users were involved in the three emails.



(a) Graph model (b) Hypergraph model

Fig. 2. Graph and hypergraph representations of Fig. 1 data. Colored hyperedges correspond to different email messages.

in three different two-user emails: Carl-Dan, Carl-Ed, and Dan-Ed. Multigraphs (i.e., graphs that allow multiple edges between pairs of vertices) are an improvement over graphs for addressing such issues. However, to accurately model multi-way (i.e., plural) relationships amongst objects with a multigraph, additional metadata (edge labels specifying which edges belong to a larger multi-way relationship) would be needed. Such an augmented multigraph model would require significant changes to the graph analysis algorithms and would offer no significant advantages over the hypergraph approaches described in the next subsection.

B. Relational Data and Hypergraph Models

While graph models can be effective in leveraging pairwise object relationships in relational data, these relationships, which often involve multiple entities, may be modeled better by hypergraphs. Hypergraphs are generalizations of graphs where edges, referred to as *hyperedges*, are more generally sets of one or more vertices (vertices still represent objects) [1]. Fig. 2b illustrates how hypergraphs can be used to model objects and their non-pairwise relationships for the email dataset of Fig. 1. Since each email expresses a relationship amongst a subset of the users, it is represented by a different

978-1-5090-3525-0/16/$31.00 ©2016 IEEE

Authorized licensed use limited to: University of Michigan Library. Downloaded on July 12,2021 at 11:29:48 UTC from IEEE Xplore. Restrictions apply.

hyperedge (represented by colored groupings of vertices). For instance, the second email is represented by the red hyperedge containing vertices corresponding to Carl, Dan, and Ed.

When modeling complex, relational data, this hypergraph model has significant advantages over its graph counterpart. As discussed for the email example, there is an explicit one-to-one mapping from relationships among groups of vertices to a hyperedge. This allows hypergraphs to account for more complex relationships than the pairwise ones modeled using graphs. Hyperedges can also be used to represent events (i.e., instances of relationships) involving different object types (e.g., people, locations, and times). This allows hypergraph models to account for temporal and locality effects in the resulting analysis. Finally, the hypergraph model can be computationally advantageous when compared to the graph model. This is evident from the incidence matrix of the hypergraph, which stores the vertices that each hyperedge contains (rows correspond to vertices, columns correspond to hyperedges, and nonzeros i, j designate that hyperedge j contains vertex i). This incidence matrix of the hypergraph requires less storage space in comparison with the graph incidence matrix that is commonly formed through the clique expansion of the hypergraph, where each hyperedge of cardinality d (i.e., containing d vertices) is expanded into $\binom{d}{2}$ graph edges, containing all pairs of the vertices in the hyperedge. For instance, the incidence matrix for the hypergraph in the email example (Fig. 2b) is:

$$H_H = \begin{pmatrix} 1 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix}$$

while the incidence matrix for the corresponding unweighted clique-expanded graph (Fig. 2a) is:

$$H_G = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}.$$

The hypergraph incidence matrix also requires significantly fewer operations than the corresponding clique-expanded graph incidence matrix when multiplied by a vector (an important operation in spectral clustering).

With this unweighted (or unit-weighted) graph model, it is also important to note that vertices that are involved in high-cardinality relationships (relationships that connect many vertices) are disproportionally connected. That is, for a given vertex, the sum of its graph edge weights with vertices in a high-cardinality relationship is larger than for vertices in lower cardinality relationships. One can attempt to correct this by weighting the resulting binary edges inversely proportionally to the cardinality on the hyperedges from which they are derived (e.g., if the cardinality of a hyperedge is d and its weight is w_h, all $\binom{d}{2}$ edges formed in the clique expanded graph are given the weight $w = w_h/(d-1)$).

C. Related Hypergraph Work

Although there have been some early hypergraph models for data science, when compared to graph models, the impact on applications has been minimal and there is not much well-established hypergraph software. Much of the hypergraph literature presents hypergraph models that are transformed to graph models followed by traditional graph analysis instead of hypergraph-based analysis. Moreover, there is a lack of theoretical hypergraph model analysis for understanding data science problems and building effective hypergraph algorithms. Furthermore, there is a lack of fundamental understanding of when hypergraphs model particular data science problems better than graphs.

Recently there has been some significant work exploring hypergraphs to address various data science problems. Bonacich, et al., used hypergraphs to model the importance of supra-dyadic (non-pairwise) relationships/events in their network analysis of attacks by Caribbean island native inhabitants on Spanish settlements in the sixteenth and seventeenth centuries [2]. Hypergraph models have been used in semi-supervised learning where label distributions are propagated along the hyperedges [3], [4]. In these models, hypergraphs are also advantageous in their flexibility for describing prior knowledge (e.g., known clusters can be represented by hyperedges). In biological systems analysis hypergraphs have been used to find optimal sets of proteins to target specific biochemical pathways in drug design, related protein functional groups, and find the minimal set of biochemical reactants to drive cascading sets of metabolic reactions [3], [5]. Hypergraph incidence matrices are often used by non-SQL databases to store data. One such example is the D4M exploded schema, which has been applied to many applications in bioinformatics, social networks, etc. and has two tables that are equivalent to hypergraph incidence matrices [6].

This paper builds on the work of Zhou, et al., who used hypergraphs for spectral clustering methods [7]. They generalized the graph spectral clustering techniques to hypergraphs and showed for some small datasets that the hypergraph methods produce better results than graph methods in the context of their machine learning applications. However, these results had some shortcomings. First, the computations take place on very small datasets. Second, it is unclear that the hypergraph is compared to the correct graph model. We assume that they use an unweighted graph model but this is unclear. (The described weighted graph model weights graph edges proportionally instead of inversely proportionally to the cardinality of the hyperedges from which they derive.) Finally, their work does not address the computational advantages of hypergraphs over graphs in the context of spectral clustering, which we have found to be the most significant advantage.

The focus of this paper is on hypergraph models of relational data and demonstrating their advantages over graph models in the context of spectral clustering, producing higher

978-1-5090-3525-0/16/$31.00 ©2016 IEEE

Authorized licensed use limited to: University of Michigan Library. Downloaded on July 12,2021 at 11:29:48 UTC from IEEE Xplore. Restrictions apply.

quality clustering with lower computational costs. We build upon the work of Zhou, et al., making a stronger argument computationally for hypergraph spectral clustering to complement their more theoretical work. The primary contributions of this paper include:

- Spectral clustering results indicating that hypergraph models, in general, produce higher quality clusterings than the graph models,
- Strong evidence that it is computationally advantageous to use hypergraphs instead of graphs for spectral clustering, and
- A new high performance analysis framework TRIDATA, which provides linear algebra based graph and hypergraph analysis, supports most HPC architectures, and targets very large datasets (e.g., billions of vertices).

II. SPECTRAL CLUSTERING

The focus of this paper is the application of hypergraphs to the clustering of relational data, which has a wide range of applications in sociology, biology, and computer science, from finding groups in social networks to finding related protein functional groups [8]. In this data clustering, the goal is to determine groupings of data objects given a set of relationships among those objects. Although relational data clustering methods have traditionally relied on graph models, this work has recently been extended to leverage hypergraph models [7]. One approach to data clustering is through spectral analysis of the graph or hypergraph, which often outperforms other approaches [9]. Below we discuss both graph and hypergraph approaches to spectral clustering.

The primary component of spectral graph clustering is the graph Laplacian matrix. Fiedler demonstrated the importance of graph Laplacians to spectral clustering, showing that bi-clustering of a graph into two groupings is closely connected with the second eigenvector of the graph Laplacian [10], now designated the Fieldler vector. It is important to note that there are many graph Laplacians that have slightly different properties. One commonly used graph Laplacian in spectral clustering is the normalized graph Laplacian:

$$L_G = I - D_{vG}^{-\frac{1}{2}}(H_G H_G^T - D_{vG})D_{vG}^{-\frac{1}{2}},$$

where H_G is the incidence matrix of the graph and D_{vG} is a diagonal matrix containing the vertex degrees [11]. To detect communities in the graph, the Fiedler vector can be recursively computed for the normalized graph Laplacian of each subcommunity. However, a common alternative procedure (which is used throughout this paper) is to (1) form the graph Laplacian for the entire graph, (2) compute several of its eigenpairs, and (3) run k-means [12] on the subspace spanned by those eigenvectors to find the specified number of communities [13].

The process of spectral hypergraph clustering is similar to that of spectral graph clustering, except that a hypergraph Laplacian is used instead of the graph Laplacian. Zhou, et al, derived such a hypergraph Laplacian [7]:

$$L_h = I - D_{vH}^{-1/2} H_H D_{eH}^{-1} H_H^T D_v^{-1/2},$$

where H_H is the incidence matrix of the hypergraph, D_{vH} is a diagonal matrix containing the vertex degrees, and D_{eH} is a diagonal matrix containing the hyperedge cardinalities. As with graph Laplacians, alternative hypergraph Laplacians have been proposed [14]. We chose to use the hypergraph Laplacian proposed by Zhou, et al., because of its nice property that for graphs (hypergraphs with 2-cardinality hyperedges) it has the same eigenvectors as the normalized graph Laplacian. This property is useful for comparison of the hypergraph and graph models, which will yield the same clustering result.

III. TRIDATA SOFTWARE

A. Spectral Clustering Functionality

We have developed a new parallel software product, Trilinos for data analytics or TRIDATA, which implements spectral methods for data clustering of relational data. TRIDATA is built on top of the the Trilinos parallel software framework [15] in order to solve these problems at a large scale (with an eventual target of up to billions of vertices). TRIDATA implements the spectral clustering procedure outlined in Section II for both the graph and hypergraph models. In TRIDATA, the graph and hypergraph Laplacian matrices are not formed explicitly. Instead, the additional diagonal matrices needed to apply the Laplacian operators: D_{vG}, D_{vH}, and D_{eH} are formed and stored. When the eigensolver needs the Laplacian operator applied to a vector (i.e., the Laplacian matrix multiplied by the vector), the operator is applied implicitly through a series of sparse matrix-dense vector multiplication operations and vector additions. The Laplacian operators are not formed explicitly for two main reasons. First, the sparse matrix-sparse matrix multiplication needed to form the Laplacian explicitly is significantly more expensive that the operations needed to implicitly apply the Laplacian operator. Thus, if the number of eigensolver iterations is small, it will be advantageous to apply the operator implicitly. Second, for dynamically changing graphs, it is easier to update the matrices used in the implicitly applied operator (incidence matrix and diagonal matrices) than to update the Laplacian matrix.

TRIDATA leverages Trilinos' efficient parallel sparse eigensolver package Anasazi [16], which implements many popular eigensolvers methods. Anasazi supports MPI+X, where X includes OpenMP, CUDA, and Pthreads; and it has been used to solve data analytics problems with over a billion vertices [17]. TRIDATA implements a hyperspherical version of k-means, where each data point of the resulting eigenvectors is normalized. The TRIDATA implementation of k-means is built on top of the Trilinos software stack, runs in parallel, and uses the k-means++ methodology for initializing the centroids given a random seed [18]. Cluster accuracy is assessed using the Jaccard index between computed spectral clustering solutions and known communities. Jaccard indices of 1.0 are defined to be a perfect match and lower values match less perfectly.

B. Hypergraph and Graph Generation

TRIDATA implements a simple Trilinos-based hypergraph generator that generates hypergraphs of different scales and an

978-1-5090-3525-0/16/$31.00 ©2016 IEEE

Authorized licensed use limited to: University of Michigan Library. Downloaded on July 12,2021 at 11:29:48 UTC from IEEE Xplore. Restrictions apply.

"ideal" vertex to cluster assignment. This hypergraph generator produces random hypergraph incidence matrices based on a variety of input parameters: the number of communities in the hypergraph, the mean number of vertices per cluster, the mean numbers of intra- and inter-cluster hyperedges, and the mean intra- and inter-cluster hyperedge cardinalities. Means are for Poisson distributions from which specific values are randomly chosen (e.g., each intra-cluster hyperedge has a cardinality assigned from this distribution).

The TRIDATA hypergraph generator is similar in principle to the stochastic block model for graphs [19], [20]: the parameters define probabilities that vertices within and across clusters will be connected by a hyperedge. Since, by construction, the number of vertices in each cluster is known and the vertices are ordered consecutively within the clusters, the cluster assignments of each of the vertices are also known. Figure 3 shows an example of a four-cluster hypergraph incidence matrix generated by the TRIDATA hypergraph generator, where each dot represents a matrix nonzero. The first four blocks of nonzeros (viewing the matrix left to right) correspond to the four clusters of vertices, with hyperedges connecting these vertices to other vertices within the same cluster. The fifth "tall" block of nonzeros corresponds to the inter-cluster hyperedges, those connecting vertices from different clusters (i.e., noise).

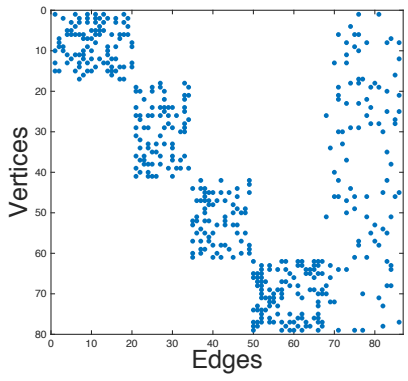


Fig. 3. Incidence matrix generated by hypergraph generator.

Graphs corresponding to a given hypergraph are constructed by expanding each hyper edge into a clique as specified in Section I-B. TRIDATA generates both unweighted and weighted graph incidence matrices, and can easily be extended to include weighting schemes other than the one used in this paper. Initial results of tests using weighted and unweighted graphs indicate that the weighted graphs result in significantly higher Jaccard indices than the corresponding unweighted graphs. Thus, experiments presented in this paper use the weighted graph model.

IV. RESULTS

This section presents results for the graph and hypergraph models in the TRIDATA spectral clustering implementations. Experiments were conducted on a 24 core workstation with 128 GB of memory (dual Intel Xeon E5-2680 v3 processors @ 2.5 GHz) using 16 MPI processes. The TRIDATA hypergraph generator is used to generate hypergraph incidence matrices for several different sets of parameters in order to better understand how effective the graph and hypergraph methods are for datasets with different structures and various amounts of noise. Table I summarizes the parameters (chosen to explore the parameter space with both strongly and weakly connected clusters) used to generate the incidence matrices. Each of the first three numerical experiments (P1, P2, P3) correspond to a specific sets of generator parameters. Since most parameters correspond to means, 10 matrices were generated for each set of parameters, and results presented are averages of experiments across those 10 instances. The fourth numerical experiment (HC) corresponds to a range of generator parameters, where we sweep over the intra-cluster and inter-cluster hyperedge cardinality parameters in the range $\{3, 4, \ldots, 10\}$.

TABLE I
DATASETS

	P1	P2	P3	HC
Number of clusters	10	5	10	5
Vertices per cluster	10000	10000	10000	10000
Intra-cluster h-edges per cluster	40000	20000	20000	20000
Inter-cluster h-edges per cluster	50000	200000	200000	200000
Intra-cluster h-edge cardinality	5	10	5	3-10
Inter-cluster h-edge cardinality	5	3	5	3-10

The Laplacian eigensystems are solved using the LOBPCG eigensolver in Anasazi. For the experiments presented here, the number of eigenpairs computed equals the number of communities The eigensolver tolerance of 10^{-3} was used, as preliminary experiments showed that lower tolerances did not change the results significantly. For each set of eigenvectors, k-means++ was run with a different initial starting state, and the best result (in terms of the k-means distance sum) was used for the final predicted communities. This multistart approach increases the likelihood of avoiding suboptimal local minima.

We ran the numerical experiments for the problems listed in Table I and compared the results and runtimes between the graph and hypergraph models.

A. Graph versus Hypergraph: Clustering Quality

Table II shows the average Jaccard indices for graphs and hypergraphs resulting from the parameters P1, P2, and P3. For these datasets, the clustering resulting from the hypergraph Laplacian method agrees very well with the clusters defined by the generator (i.e., the ground truth), with Jaccard indices close to 1.0. The clustering from the graph Laplacian method, however, varies in its agreement with the ground truth. The Jaccard indices for P3 and P2 were 0.967 and 0.590, respectively. However, the general trend is that the hypergraph results are more consistent with the ground truth than the graph results.

There is an interesting trend when comparing the spectral clustering for the graph and hypergraph Laplacians for the range of parameters in experiment HC. Fig. 4 shows a

978-1-5090-3525-0/16/$31.00 ©2016 IEEE

Authorized licensed use limited to: University of Michigan Library. Downloaded on July 12,2021 at 11:29:48 UTC from IEEE Xplore. Restrictions apply.

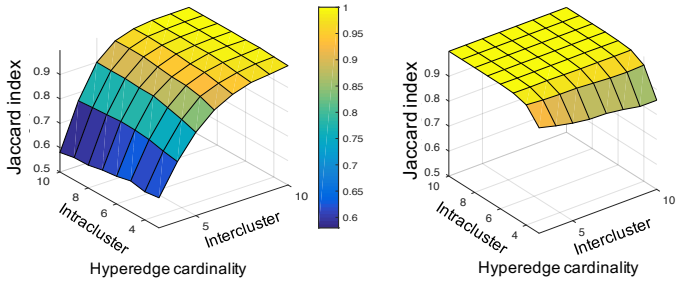
TABLE II
GRAPH VERSUS HYPERGRAPH: JACCARD INDICES

	Graph	Hypergraph
P1	1.000	1.000
P2	0.590	1.000
P3	0.967	0.999

TABLE III
GRAPH VERSUS HYPERGRAPH: SPECTRAL CLUSTERING RUNTIMES

	Graph (seconds)	Hypergraph (seconds)	Ratio (G/H)
P1	9.67	1.73	5.6
P2	12.82	0.42	30.5
P3	14.60	2.40	6.1

plot of the the Jaccard indices for graphs and hypergraphs for the parameters in this range. The overall trend is that the hypergraph results are more consistent with the ground truth than the graph results, in general. For graphs derived from hypergraphs with low average inter-cluster hyperedge cardinality, the agreement with the ground truth is fairly poor (≈ 0.6). However, for hypergraphs with low average intra-cluster hyperedge cardinality, there is a significant decline in clustering performance, including a small region where the hypergraph model deviates more from the ground truth than the graph model. This trend in the hypergraph data is consistent with what may be expected using noisy data, since increasing the intra-cluster cardinality will make the clusters easier to identify. The trend in the graph data (increasing inter-cluster cardinality resulting in cluster identification more consistent with the ground truth), however, is unexpected, and may be related to the low Jaccard index for P2. We hypothesize that this is a shortcoming of the weighting of the graph model and a more complicated weighting scheme for this graph model is necessary to be effective for these problems.



(a) Graph model (b) Hypergraph model

Fig. 4. Jaccard indices for the HC hypergraphs.

B. Graph versus Hypergraph: Runtime

Table III shows the average runtimes for spectral clustering of the graphs and hypergraphs resulting from the parameters P1, P2, and P3. It is important to note that the graph runtimes do not include the additional time necessary for the clique expansion, which would be an additional cost. For these datasets, the spectral clustering method using the hypergraph Laplacian performs significantly more efficiently than the method using the graph Laplacian, with up to a factor of 30 reduction in the runtime.

A similar trend exists for the experiments comparing the spectral clustering for the graph and hypergraph Laplacians for the range of parameters HC. Figure 5 shows the ratio of the runtimes of the spectral clustering using the graph Laplacian to the runtimes of the spectral clustering using the hypergraph Laplacian for each of the parameters in this range. The runtime of the spectral clustering using the hypergraph Laplacian is significantly less than when using the graph Laplacian, ranging from a factor of 4 up to a factor of 30. The overall trend is that hypergraph has an increasing computational advantage as the hyperedge cardinalities are increased, which is expected because an increase in cardinality results in an increase in the number of graph edges created during the clique expansion process. An unexpected trend illustrated here is that for an intra-cluster hyperedge cardinality of 10, this runtime ratio decreases as the inter-cluster hyperedge cardinality is increased.

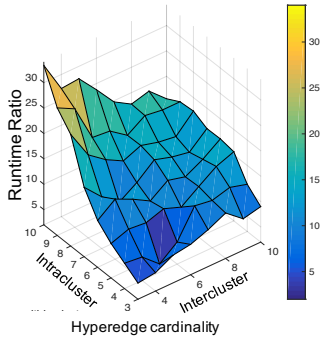


Fig. 5. Ratio of graph to hypergraph runtimes for HC hypergraphs.

The number of iterations required by the eigensolver and k-means for convergence (shown in Fig. 6) explains some of the computational advantages that we see for the hypergraph over the graph. The eigensolver, which is the most costly step in the spectral clustering process, takes up to 4.8 times fewer iterations for the hypergraph model than it does for the graph model. This is a result of the hypergraph Laplacian having a more favorable eigenvalue spectrum (i.e., better separation of eigenvalues) than that of the graph Laplacian for these problems. Fig. 7 illustrates a similar trend for the HC matrices, where LOBPCG for the hypergraph model can take up to six times fewer iterations. The number of LOBPCG iterations are particularly high for graphs derived from high intra-cluster hyperedge cardinality and low inter-cluster hyperedge cardinality, which explains the previously mentioned anomaly seen with the runtime ratio decreasing as the inter-cluster cardinality is increased for problems with intra-cluster hyperedge cardinality of 10.0. Running k-means takes up to 11 times fewer iterations to converge for the eigenvectors resulting from the hypergraph Laplacian when

978-1-5090-3525-0/16/$31.00 ©2016 IEEE

Authorized licensed use limited to: University of Michigan Library. Downloaded on July 12,2021 at 11:29:48 UTC from IEEE Xplore. Restrictions apply.

compared to the graph Laplacian (although k-means runtimes are insignificant relative to eigensolver runtimes).

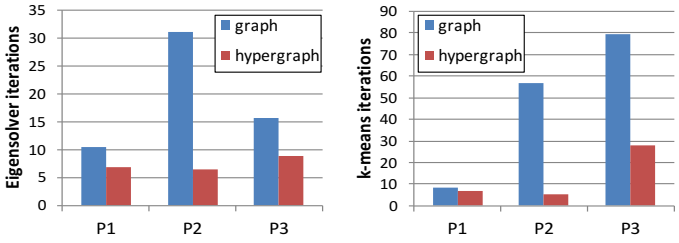
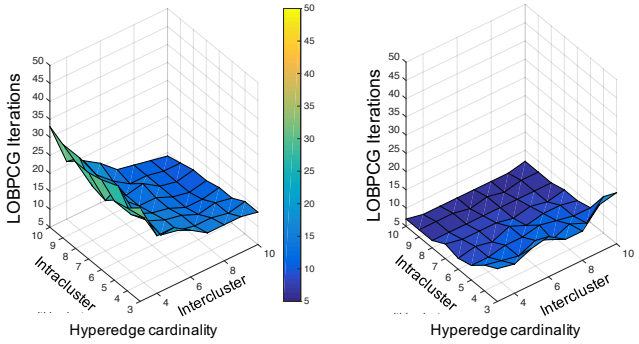


Fig. 6. Average number of LOBPCG and k-means iterations for graph and hypergraph spectral clustering for the P1, P2, and P3 hypergraphs.



(a) Graph model (b) Hypergraph model

Fig. 7. Average number of LOBPCG iterations for graph and hypergraph spectral clustering for the HC hypergraphs.

Another factor contributing to the discrepancy between the runtimes of the spectral graph and hypergraph clustering is the time required to apply the Laplacian operators to vectors as part of the eigensolver. Table IV and Fig. 8 show that the application of the hypergraph Laplacian is significantly less costly than that of the graph Laplacian, with up to a factor of 17 reduction in runtime. As mentioned in Subsection I-B, multiplying the hypergraph incidence matrix by a vector is much less costly than multiplying the clique-expanded graph incidence matrix by a vector. Since this is a major component of the computation in the application of the Laplacian operators, it makes sense that the hypergraph Laplacian is more efficient to apply than the graph Laplacian.

TABLE IV
GRAPH VERSUS HYPERGRAPH: TIME TO APPLY LAPLACIAN OPERATOR

	Graph (seconds)	Hypergraph (seconds)	Ratio (G/H)
P1	0.6684	0.0847	7.9
P2	0.3620	0.0297	12.1
P3	0.6755	0.0885	7.6

V. SUMMARY/CONCLUSIONS

In this paper, we examined graph versus hypergraph models of relational data in a very specific context, spectral clustering of relational data, which has many important applications. To

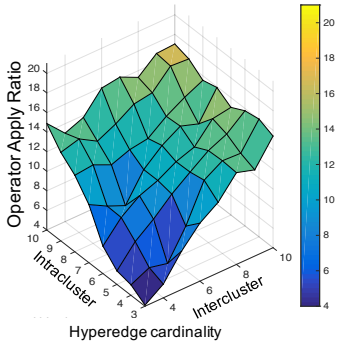


Fig. 8. Ratio of time for applying graph Laplacian operator to time for applying the hypergraph Laplacian operator for HC hypergraphs.

drive applications in this area, we have developed TRIDATA, a high performance analysis framework that supports spectral clustering and many different HPC architectures. Although the clustering quality results were complicated, the general trend was that the hypergraph model produced results more similar to the generator's ground truth than what was produced by the graph model, sometimes with some significant differences. In particular, it was difficult to find a weighting for the graph model (e.g., based on the original hyperedge cardinalities) that yielded good quality clustering across the datasets. In addition, we showed strong evidence that the hypergraph based methods were computationally superior to their graph counterparts, with up to a factor of 30 reduction in runtime.

In future work, we plan to continue exploring hypergraph applications to data science problems. We will expand our spectral clustering work to study real datasets and further understand the impact of hypergraphs. We plan to develop better theoretical hypergraph models, which are lacking when compared to the numerous graph models, and hypergraph algorithms inspired by those models. With these significant breakthroughs, hypergraphs promise to fundamentally change the way people analyze relational data as part of decision-making. In addition, we plan to further develop TRIDATA, which is already a powerful platform for analyzing graphs and hypergraphs using high performance linear algebra techniques. We plan to improve the performance of the hypergraph generator and integrate 2D partitioning techniques [21]–[23], which will improve scalability of our approach and allow us to analyze graphs and hypergraphs of up to billions of vertices. Finally, we plan to extend the hypergraph models in TRIDATA to target other algorithms (e.g., eigencentrality, information propagation), which would allow us to use this powerful tool to impact additional data science applications.

ACKNOWLEDGMENT

We thank Richard Lehoucq and Jon Berry for helpful discussions. 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.

978-1-5090-3525-0/16/$31.00 ©2016 IEEE

Authorized licensed use limited to: University of Michigan Library. Downloaded on July 12,2021 at 11:29:48 UTC from IEEE Xplore. Restrictions apply.

REFERENCES

[1] C. Berge, *Hypergraphs: combinatorics of finite sets*. Elsevier, 1984, vol. 45.

[2] P. Bonacich, A. C. Holdren, and M. Johnston, "Hyper-edges and multidimensional centrality," *Social Networks*, vol. 26, p. 189203, July 2004.

[3] K. Tsuda, "Propagating distributions on a hypergraph by dual information regularization," in *Proceedings of the 22nd international conference on Machine learning*. ACM, 2005, pp. 920–927.

[4] A. Corduneanu and T. Jaakkola, "Distributed information regularization on graphs," in *Advances in Neural Information Processing Systems 17: Proceedings of the 2004 Conference*, vol. 17. MIT Press, 2005, p. 297.

[5] S. Klamt, U.-U. Haus, and F. Theis, "Hypergraphs and cellular networks," *PLoS Comput Biol*, vol. 5, no. 5, p. e1000385, 2009.

[6] J. Kepner, C. Anderson, W. Arcand, D. Bestor, B. Bergeron, C. Byun, M. Hubbell, P. Michaleas, J. Mullen, D. O'Gwynn *et al.*, "D4m 2.0 schema: A general purpose high performance schema for the accumulo database," in *High Performance Extreme Computing Conference (HPEC), 2013 IEEE*. IEEE, 2013, pp. 1–6.

[7] D. Zhou, J. Huang, and B. Schölkopf, "Learning with hypergraphs: Clustering, classification, and embedding," in *Advances in neural information processing systems*, 2006, pp. 1601–1608.

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

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

[10] M. Fiedler, "Algebraic connectivity of graphs," *Czechoslovak mathematical journal*, vol. 23, no. 2, pp. 298–305, 1973.

[11] F. R. Chung, *Spectral graph theory*. American Mathematical Soc., 1997, vol. 92.

[12] S. P. Lloyd, "Least squares quantization in pcm," *Information Theory, IEEE Transactions on*, vol. 28, no. 2, pp. 129–137, 1982.

[13] A. Y. Ng, M. I. Jordan, Y. Weiss *et al.*, "On spectral clustering: Analysis and an algorithm," *Advances in neural information processing systems*, vol. 2, pp. 849–856, 2002.

[14] F. Chung, "The laplacian of a hypergraph," *Expanding graphs (DIMACS series)*, pp. 21–36, 1993.

[15] M. A. Heroux, R. A. Bartlett, V. E. Howle, R. J. Hoekstra, J. J. Hu, T. G. Kolda, R. B. Lehoucq, K. R. Long, R. P. Pawlowski, E. T. Phipps, A. G. Salinger, H. K. Thornquist, R. S. Tuminaro, J. M. Willenbring, A. Williams, and K. S. Stanley, "An overview of the Trilinos project," *ACM Trans. Math. Softw.*, vol. 31, no. 3, pp. 397–423, 2005.

[16] 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, pp. 13:1–13:23, Jul. 2009. [Online]. Available: http://doi.acm.org/10.1145/1527286.1527287

[17] M. M. Wolf and B. A. Miller, "Sparse matrix partitioning for parallel eigenanalysis of large static and dynamic graphs," in *High Performance Extreme Computing Conference (HPEC), 2014 IEEE*. IEEE, 2014, pp. 1–6.

[18] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," in *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics, 2007, pp. 1027–1035.

[19] S. E. Fienberg and S. Wasserman, "Categorical data analysis of single sociometric relations," *Sociological methodology*, vol. 12, pp. 156–192, 1981.

[20] P. W. Holland, K. B. Laskey, and S. Leinhardt, "Stochastic blockmodels: First steps," *Social networks*, vol. 5, no. 2, pp. 109–137, 1983.

[21] A. Yoo, A. H. Baker, R. Pearce *et al.*, "A scalable eigensolver for large scale-free graphs using 2d graph partitioning," in *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*. ACM, 2011, p. 63.

[22] E. G. Boman, K. D. Devine, and S. Rajamanickam, "Scalable matrix computations on large scale-free graphs using 2d graph partitioning," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*. ACM, 2013, p. 50.

[23] M. M. Wolf and B. A. Miller, "Sparse matrix partitioning for parallel eigenanalysis of large static and dynamic graphs," in *High Performance Extreme Computing Conference (HPEC), 2014 IEEE*. IEEE, 2014, pp. 1–6.

978-1-5090-3525-0/16/$31.00 ©2016 IEEE

Authorized licensed use limited to: University of Michigan Library. Downloaded on July 12,2021 at 11:29:48 UTC from IEEE Xplore. Restrictions apply.