Relaxation-Based Coarsening for Multilevel Hypergraph Partitioning

Ruslan Shaydulin
School of Computing
Clemson University
rshaydu@clemson.edu

Jie Chen
IBM Thomas J. Watson
Research Center
chenjie@us.ibm.com

Ilya Safro
School of Computing
Clemson University
isafro@clemson.edu

January 11, 2019

Abstract

Multilevel partitioning methods that are inspired by principles of multiscaling are the most powerful practical hypergraph partitioning solvers. Hypergraph partitioning has many applications in disciplines ranging from scientific computing to data science. In this paper we introduce the concept of algebraic distance on hypergraphs and demonstrate its use as an algorithmic component in the coarsening stage of multilevel hypergraph partitioning solvers. The algebraic distance is a vertex distance measure that extends hyperedge weights for capturing the local connectivity of vertices which is critical for hypergraph coarsening schemes. The practical effectiveness of the proposed measure and corresponding coarsening scheme is demonstrated through extensive computational experiments on a diverse set of problems. Finally, we propose a benchmark of hypergraph partitioning problems to compare the quality of other solvers.

1 Introduction

Hypergraphs are generalizations of graphs. Both graphs and hypergraphs are ordered pairs of sets (V, E), where V is the vertex set and E is the set of (hyper)edges such that each $e \in E$ is a subset of vertices. The difference is that in a graph, the cardinality of each edge is exactly two, whereas in a hypergraph, a hyperedge can contain an arbitrary number of vertices. The hypergraph partitioning (HGP) problem is therefore a generalization of the graph partitioning (GP) problem. In GP the goal is to split the set of vertices into multiple sets (usually called partitions) of similar sizes, such that a cut metric is minimized. Here, a cut defines a set of (hyper)edges spanning more than one partition. In the HGP generalization, the hyperedges can possibly span more than two partitions. There exist several versions of minimization objectives, constraints, and cut metrics in both GP and HGP [5, 10]. Hypergraph partitioning has many applications, including VLSI design [33, 2], parallel matrix multiplication [12], classification [55], cluster ensembling [51], and combinatorial scientific computing [41], among others [5, 43].

1.1 Multilevel partitioning

One of the most popular and fastest approaches for solving HGP problems is multilevel: first, the hypergraph is iteratively coarsened by merging its vertices; then, the solution (i.e., the partitioning)

is computed at the coarsest level; and finally, the coarsest solution is gradually uncoarsened back to the finest level (see fig. 1). Coarsening consists of two steps, namely, aggregation and contraction. During the aggregation step, the decision is made on which vertices to merge. At the contraction step, these vertices are merged and hyperedges are dropped. The reverse process, uncoarsening, also consists of two steps, namely, interpolation and refinement. In the interpolation step, the solution for coarse vertices (i.e., the assignment of vertices into corresponding partitions) is interpolated into the next-fine level. The refinement step improves the interpolated solution, typically, by iteratively solving a sequence of local optimization problems that refine the objective or constraints of HGP. In many state-of-the-art solvers, such refinement is implemented using vertex-moving heuristics that accommodate a vertex in a better partition.

The multilevel method captures the global structure of a hypergraph in the solution by combining local information at different scales of coarseness. The aims of coarsening are (a) preserving the structural properties of the original hypergraph throughout the entire multilevel hierarchy until the smallest hypergraph is obtained, and (b) allowing an effective interpolation in which the solution of coarse level variables is correctly projected into the corresponding fine level ones. Typically, a high-quality coarsening should also preserve the spectral properties of the (hyper)graph adjacency matrix (even though it is not directly required by a coarsening algorithm). The coarsest hypergraph is partitioned, and the solution is projected on larger, finer-level hypergraphs all the way back to the original one. At each level the projected solution is refined.

As a result, the multilevel method ends up with a solution that is aware of the global structure and is locally optimal as well. Note, however, that the global optimality of the final solution heavily depends on the quality of coarsening. Therefore, it is paramount to have a good distance/similarity measure on vertices, one that captures not only how vertices are similar *locally*, but also how close they are *globally*. In other words, a sophisticated similarity measure for vertices (and a resulting more accurate aggregation) allows for a more accurate interpolation of the solution in the next step.

In most cases, multilevel algorithms for HGP are generalizations of those for GP and other cut optimization problems; they are, in turn, simplified adaptations of multigrid and other multiscale optimization strategies [7]. There exist a few coarsening methods for different cut problems on graphs; e.g., various versions of weighted aggregations inspired by algebraic multigrid for graph reordering [45, 47, 44, 29], and matching-based methods for GP [33, 48], vertex separator problem [25], and clustering [20]. For GP, advanced similarity measures used in coarsening include flows [17], diffusion [36], and algebraic distances [46]. For an in-depth overview, the reader is referred to a survey by Buluç et al. [10]. However, to the best of our knowledge there exist few advanced similarity concepts for coarsening applied to HGP.

1.2 Our contribution

We introduce a similarity measure for vertices of hypergraphs, as well as a relaxation-based coarsening scheme for multilevel hypergraph partitioning. The measure is called *algebraic distance for hypergraphs*. It generalizes the algebraic distance developed for graphs [15, 44], used in applications such as graph partitioning [46], sparsification [31], and other problems [47, 38]. The algebraic distance takes into account distant neighborhoods of vertices and addresses the issue of unweighted and noisy hyperedges. At each level of coarsening, we compute algebraic distances on a bipartite representation of the hypergraph and modify the hyperedge weights that are used for agglomerative inner-product matching [13]. This way, we penalize cutting hyper-
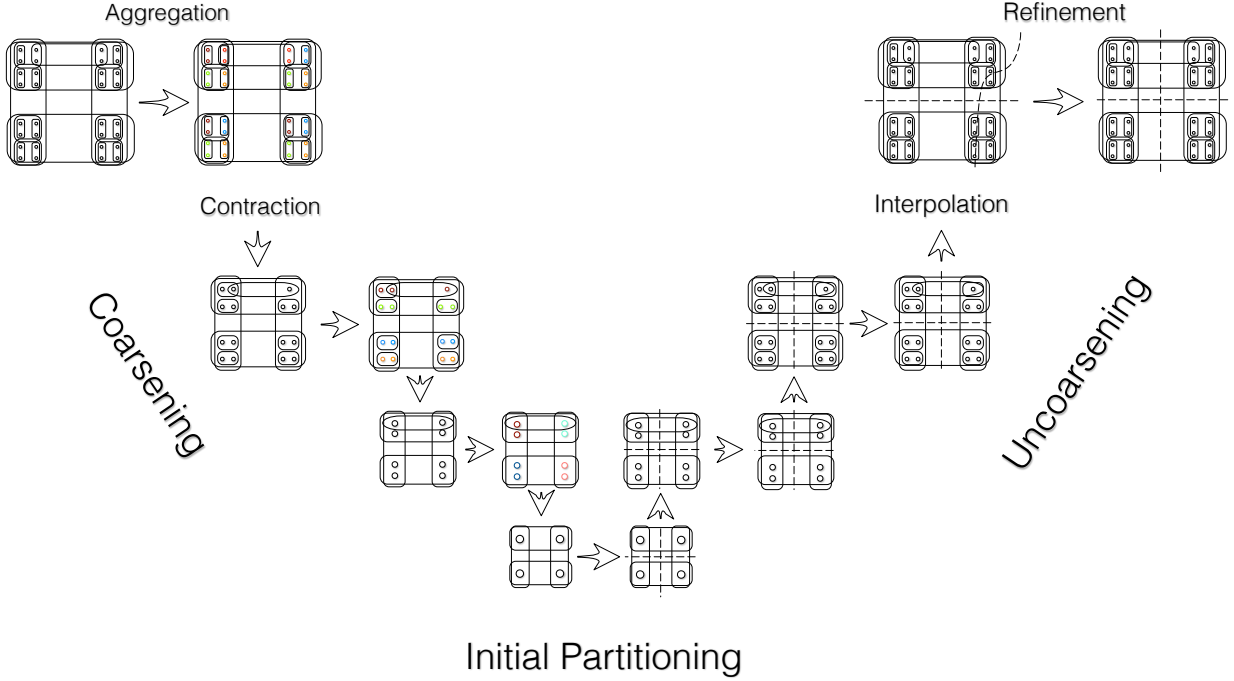
Figure 1: Outline of multilevel hypergraph partitioning scheme. On the left side of the V-cycle (named after the shape of the coarsening-uncoarsening hierarchy), the hypergraph is iteratively coarsened by aggregating (horizontal arrows) and contracting (vertical arrows) vertices. On the right side of the V-cycle, the initial partitioning is repeatedly interpolated (vertical arrows) into the next-fine level and refined (horizontal arrows).

edges that contain similar vertices measured by the algebraic distances between them. Our implementation is available at `https://github.com/rsln-s/algebraic-distance-on-hypergraphs`. We also propose a comprehensive benchmark of hypergraphs and the partitioning results for a broad scientific community to evaluate the quality of their solvers. Our benchmark is available at `http://shaydul.in/hypergraph-partitioning-archive/`.

2 Preliminaries

A hypergraph is an ordered pair of sets (V, E), where V is the set of vertices and E is the set of hyperedges. Each hyperedge is a subset of V. Another terminology is also used in the literature, wherein a hypergraph is defined as a triplet (V, H, P), where V is the set of vertices, H is the set of hyperedge labels and P is the set of pins. Each pin connects a vertex from V with a label from H. A net (a synonym of hyperedge) is therefore a set of pins connecting the vertices to the corresponding label [5]. In this paper, we will use only the former terminology. We will also refer to a hyperedge as simply an "edge" when it does not cause confusion. Both vertices and hyperedges can have weights, namely, $w(v) \in \mathbb{R}_{\geq 0}$ and $w(e) \in \mathbb{R}_{\geq 0}$ respectively for each $v \in V, e \in E$. A zero weight for a hyperedge assumes that it is not present in the hypergraph.

3

2.1 Hypergraph partitioning

Hypergraph partitioning is a generalization of graph partitioning. The aim is to partition the set of vertices into a number of disjoint sets such that a cut metric is minimized subject to some imbalance constraint. Partitioning into two parts is often referred to as *bipartitioning*, and partitioning into k parts, *k-way partitioning*.

More formally, if V is the set of vertices and E is the set of edges, a *partitioning* is a set of k mutually disjoint subsets $V_i \subseteq V$, $1 \leq i \leq k$, such that $V_1 \cup V_2 \cup .. \cup V_k = V$. Then, the cut is defined as $E_{cut} \subset E$ such that for $e \in E$ holds $e \in E_{cut}$ if and only if $\exists i, j \in \mathbb{N}$, $i \neq j$: $e \cap V_i \neq \emptyset, e \cap V_j \neq \emptyset$.

Several cut metrics have been introduced over the years [32, 2, 12] for HGP. Two of the most commonly used are *hyperedge cut* and *connectivity*. The hyperedge cut cost is equal to the total weights of edges that span more than one partition, i.e.,

$$\sum_{e \in E_{cut}} w(e).$$

The connectivity objective is defined as

$$\sum_{e \in E_{cut}} (\lambda(e) - 1),$$

where $\lambda(e)$ is the number of partitions spanned by the hyperedge e [18]. In this paper we aim at minimizing the hyperedge cut metric.

The imbalance constraint specifies to what extent the partitions can differ in their size. In this paper we will use the imbalance constraint defined in the state-of-the-art hypergraph partitioner Zoltan [18]. Imbalance is defined as the sum of vertex weights in the maximum partition over the average sum of vertex weights in a partition. For k-way partitioning:

$$imbal = \frac{\sum_{v \in V_{max}} w(v)}{\frac{1}{k} \sum_{v \in V} w(v)},$$

where V_{max} is the largest partition (i.e. $\sum_{v \in V_{max}} w(v) = \max_i \left(\sum_{v \in V_i} w(v) \right)$).

2.2 Multilevel method

The multilevel method for graph partitioning was initially introduced to speed up existing algorithms [3], but was quickly recognized as a good way to improve the quality of partitioning [34, 26]. It is used as a global suboptimal heuristic framework [10], in which other heuristics are incorporated at different stages. These three stages are coarsening, initial partitioning, and uncoarsening.

During the coarsening stage a hypergraph $H = (V, E)$ is approximated via a series of successively smaller hypergraphs $H^i = (V^i, E^i)$, $1 \leq i \leq l$, where l is the number of levels in the hierarchy. The superscript denotes the number of the corresponding level for hypergraphs, nodes, and edges, respectively. Each next-coarse hypergraph is constructed by merging or aggregating vertices in the previous one according to some heuristic: $v_k^i = \{v_{k_1}^{i-1}, ..., v_{k_j}^{i-1}\}$. That is, a vertex v_k^i in the coarse hypergraph G^i at the ith level is created by grouping a set of vertices $v_{k_1}^{i-1}, ..., v_{k_j}^{i-1}$ from the finer hypergraph H^{i-1}. Vertices can be grouped by using different criteria, with the aim of interpolating solution from coarse level nodes to the corresponding fine level nodes with minimum loss of solution

quality. In the case of pairwise grouping of vertices, such a coarsening is referred to as matching. The weight of the new coarse vertex is equal to the sum of weights of the merged vertices:

$$w(v_k^i) = w(v_{k_1}^{i-1}) + ... + w(v_{k_j}^{i-1}).$$

A coarse vertex is contained in all hyperedges that contain the merged vertices. Hyperedges of cardinality one are discarded. Coarsening terminates when the size of the hypergraph is below a certain threshold or when a solution is easy to compute.

In the partitioning stage, the coarsest hypergraph H^l is partitioned using exact or approximate solver. In many existing implementations, the solver is a local search heuristic. This partitioning is anticipated to approximate the *global* solution in the sense that it incorporates the global structure of the hypergraph. In some cases, when H^l is sufficiently small, an exact solution can be computed.

The uncoarsening stage consists of two steps, namely, interpolation and refinement. During uncoarsening, the partitioning from the coarse hypergraph H^{i+1} is projected onto the fine H^i (interpolation) and refined using a local search heuristic such as Kernighan-Lin (KL) [35] or Fiduccia-Mattheyses (FM) [21] (refinement). This retains the global information of the partitioning of the coarse hypergraph while optimizing it locally. Typically, solving a local search subproblem only improves the global solution at the same level.

3 Related work

Because HGP is NP-complete [23], many heuristics and approximations have been developed. The most common practical approach to HGP is the multilevel framework. This section begins with a brief description of non-multilevel techniques, followed by multilevel ones.

3.1 Spectral methods

An important family of non-multilevel techniques is spectral methods. It is necessary to point out that while they can be used as standalone methods, they are also often used within the multilevel framework. As we discussed in section 2.2, the multilevel method for combinatorial optimization problems is a heuristic that incorporates other heuristics as well, such as different similarity concepts and iterative refinement techniques. Spectral hypergraph partitioning generalizes spectral graph partitioning methods to hypergraphs. It usually utilizes the spectral properties of the adjacency matrix. Two main approaches are identified.

The first one is to construct a graph from the hypergraph [24] and then apply spectral graph partitioning methods that are more well-developed [42, 22]. Two of the most common approaches are star and clique expansions. In the case of clique expansion, a hyperedge is replaced by a set of edges that form a complete subgraph for the vertices in the hyperedge. In the case of star expansion, a hyperedge is replaced by a new vertex, which is connected by new edges to all vertices previously contained in the hyperedge.

This approach suffers from an obvious loss of information: when a hyperedge is expanded (i.e., replaced by a clique or a star), its vertices are connected by a number of edges. The information that they are equal members of a hyperedge is lost. Ihler et al. show [30] that even for bi-partitioning there exists no min-cut graph model of a hypergraph. That is, one cannot create a graph whose edge cut is equal to the hyperedge cut in the original hypergraph, if negative weights are not allowed [30]. Finally, the hypergraph-to-clique conversion greatly increases the size of the problem. Nevertheless,

we point out that despite these limitations, good practical results can still be obtained by using graph models of a hypergraph.

The second approach is to build hypergraph Laplacian and to study its properties, bypassing the graph representation. This can be done in various ways. Bolla defines an unweighted hypergraph Laplacian matrix and shows the link between its spectral properties and the hypergraph cut [6]. Zhou et al. define a Laplacian matrix and show a way to use it for k-way partitioning [55]. Hu et al. argue that Laplacian tensors naturally extend the graph Laplacian matrices to hypergraphs. They describe a Laplacian tensor for an even uniform hypergraph and define algebraic connectivity for it [28]. Chan et al. define a Laplacian operator induced by a stochastic diffusion process on the hypergraph and generalize Cheeger's inequality for it [14]. However, these recent advances of the spectral approaches, while promising, are not yet well developed for large-scale instances.

3.2   Multilevel methods

Most state-of-the-art hypergraph partitioners (such as hMetis2 [33], PaToH [11], Zoltan [18] and Mondriaan [54], to name a few) use a multilevel approach inspired by simplified multigrid and principles of multiscale computing.

In the coarsening stage of the V-cycle, most hypergraph partitioners use, with some variations, a heuristic that greedily aggregates neighboring vertices, with some preference based on a similarity metric. These similarity metrics are local and usually very simple. The metric used by Mondriaan [54], hMetis2 [33] and Zoltan [18] is *inner product matching*. The inner product of two vertices is defined as the Euclidean inner product of their hyperedge incidence vectors [18]. In other words, the inner product of two vertices is the number of hyperedges they have in common or, in the weighted case, total weight of those hyperedges. PaToH uses a different metric as a default option, called *absorption matching*:

$$\text{am}(u,v) = \sum_{e=(u,v)\in E} \frac{1}{|e|-1},$$

where $|e|$ is the cardinality of an edge e. However, there are scenarios where these simple solutions, while computationally efficient and easy to implement, are not very effective and can be improved [15]. This is the major reason for us to revise the coarsening strategy for hypergraphs. In the refinement step, all of the aforementioned partitioners use some variation of Fiduccia-Mattheyses [21] or Kernighan-Lin [35] including their advanced efficient implementation introduced in [1].

However, there is relatively little research on how to improve the coarsening of a hypergraph. Existing research was motivated mainly by the intuition that a decision made earlier in the coarsening stage has substantial influence on the quality of the final cut: any error or wrong decision would be propagated all the way down the V-cycle and accumulate. Some methods that improve the coarsening include a use of rough set theory [39], as well as some variations on the greedy heavy matching scheme. A very promising but unfinished attempt to generalize HGP coarsening using algebraic multigrid was published in Sandia Labs Summer Reports [9]. Another extension of a multilevel method, namely, the n-level recursive bisection, was introduced in [49]. For a more extensive review of hypergraph partitioning algorithms the reader is referred to [53].
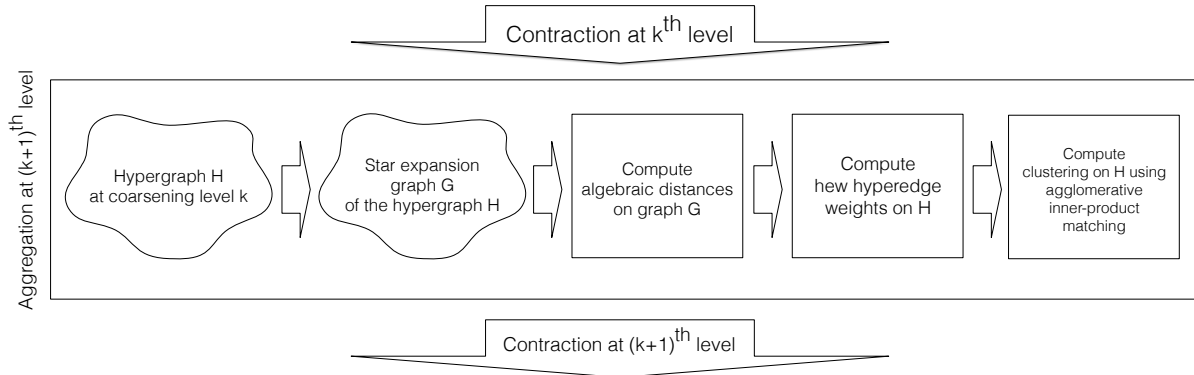
Figure 2: Outline of the algorithm used to aggregate vertices at kth coarsening level.

4 Algebraic Distance on Hypergraphs

In this section, we introduce a new distance measure that plays a crucial role in improving the quality of coarsening. To demonstrate its effectiveness, we use Zoltan [18] as the baseline solver.

The outline of our approach is as follows. At each level of coarsening, we compute new weights for the hyperedges. These weights are passed to Zoltan's coarsening subroutine, allowing it to use this additional information for making matching decisions. After the matching is computed, the weights are set back to the original hyperedge weights and the multilevel algorithm continues. In other words, we leverage the hyperedge weights to pass information on the structure of the hypergraph, derived by our algorithm, to the HGP solver's coarsening scheme. We refer to these weights as *algebraic weights*. The outline is shown in fig. 2.

Discussion of this algorithm would not be complete without a brief description of the coarsening scheme used by Zoltan. Zoltan uses an agglomerative matching technique known as inner-product matching [13], or called heavy-connectivity clustering in PaToH [12]. In this technique, the vertices are visited in random order. If the visited vertex v is unmatched, an adjacent vertex u with the highest *connectivity* is selected and the current vertex is added to its cluster C_u. The *connectivity* is defined as $N_{v,C_u}/W_{v,C_u}$, where N_{v,C_u} is the total weight of the edges connecting v with the vertices in the cluster C_u, and W_{v,C_u} is the total weight of the vertices in the candidate cluster [12].

4.1 Algorithm

Recall that the hypergraph is denoted as $H = (V, E)$. We call the star expansion graph G. In general, we use primed variables when referring to the elements of G and non-primed for the elements of H. At each coarsening level the following happens:

1. Build the star expansion graph $G = (V', E')$ of H [52]. Each hyperedge is replaced by a new vertex, which is connected to every vertex contained in the original hyperedge, i.e., $V' = V \cup E$ and $E' = \{(v, h) \mid v \in h, h \in E\}$ [52]. This way, each hyperedge is replaced by a "star" in the bipartite graph G, with all vertices $v \in V$ in one part and hyperedges $h \in E$ in another (hence the new set of vertices is $V' = V \cup E$). The weights of the vertices stay the same. The vertices introduced to replace hyperedges are assigned the weights of the corresponding hyperedges they

represent, divided by the number of vertices in that hyperedge; i.e.,

$$w'(v') = w(v), \quad \forall v \in V \subset V', \text{ and}$$

$$w'(h') = \frac{w(h)}{|h|}, \quad \forall h \in E \subset V'. \tag{1}$$

The edges $e' \in E'$ of the star expansion G are left unweighted (we can assume they all have weight 1). We refer to the vector of weights in the graph G as $\mathbf{w}' \in \mathbb{R}^{|V'|}$ and that in the hypergraph H as $\mathbf{w} \in \mathbb{R}^{|V|+|E|}$, to distinguish the two cases. Note that since $V' = V \cup E$, these vectors are of the same size.

2. Compute algebraic coordinates of the vertices in star expansion graph G by using Jacobi over-relaxation (JOR, a stationary iterative relaxation), beginning with random initialization. The iterative process is repeated for several random initial vectors.

   For each random vector, all coordinates are initialized with random values from the uniform distribution on $(-0.5, 0.5)$. We denote the algebraic coordinates as a 2-dimensional array X, where $x[r][v']$ is the algebraic coordinate for the vertex $v' \in V' = V \cup E$ and the rth random initial vector. Then, for a certain number of iterations (denoted in pseudocode as num_iter), JOR is performed on all $v' \in G$ as follows. The vertices are visited in random order. After each iteration, the algebraic coordinates are rescaled such that the smallest algebraic coordinate is equal to -0.5 and the largest to 0.5. The iteration scheme can be written as follows

$$x^*[r][v'] = \frac{\sum\limits_{\forall u' \in V':(u',v') \in E'} w'(u') \cdot x^{i-1}[r][u']}{\sum\limits_{\forall u' \in V':(u',v') \in E'} w'(u')} \tag{2}$$

$$x^{**}[r][v'] = \omega \cdot x^*[r][v'] + (1 - \omega)x^{i-1}[r][u'],$$

   where ω is a relaxation factor. It is used in the same way as in Successive overrelaxation [8], to make the convergence more stable. Weighting, rescaling and the denominator in eq. (2) are introduced in part to prevent algebraic coordinates from converging to machine precision, i.e., becoming so close that they are no longer distinguishable. In our experiments, nondistinguishable coordinates tend to occur on hypergraphs whose hyperedges have high cardinality, since vertices contained in these hyperedges very quickly "pull" each other together.

   Denote the smallest ("leftmost" on the real line) algebraic coordinate before rescaling as l and the largest ("rightmost") as r. Then, the rescaling is done as follows:

$$x^i[r][v'] = \frac{x^{**}[r][v'] - l}{r - l} - 0.5.$$

3. Define the algebraic weight of a hyperedge $h \in E$ as one over the algebraic distance between two farthest apart vertices in h, maximized over all random vectors:

$$alg\_weight(h) = \frac{1}{\max\limits_{\forall r} \max\limits_{u,v \in h} |x[r][v] - x[r][u]|}.$$

8

Compute the final weights $\tilde{\mathbf{w}}$ to be passed to Zoltan from the original hyperedge weights $\mathbf{w}$ as follows

$$\tilde{w}(h) = w(h) \cdot \frac{alg\_weight(h)}{\displaystyle\sum_{h^* \in E} alg\_weight(h^*)/|E|}.$$

Hyperedge weights are multiplied by the ratio between the computed algebraic distance for this hyperedge and the average algebraic weight for all hyperedges. Note that here $w(h)$ is the weight of the original hyperedge, before the star expansion.

4. Pass these new weights $\tilde{\mathbf{w}}$ to Zoltan's [18] agglomerative inner product matching. After the matching, all weights are reset back to the ones before the star expansion (i.e., back to $\mathbf{w}$) and the multilevel process continues.

The pseudocode for computing the algebraic weights (steps 1–3) is presented in algorithm 1.

ALGORITHM 1: Computing algebraic weights

Input : Relaxation factor ω, number of random vectors R , number of iterations num_iter
Output: algebraic weights alg_weight

for $r = 1 : R$ do
    Randomly initialize $x[r]$;
    for $k = 1 : num\_iter$ do
        // Perform iteration sweep over all vertices
        for $v' \in V'$ do

$$x^*[r][v'] = \frac{\displaystyle\sum_{\forall u' \in V':(u',v') \in E'} w'(u') \cdot x^{i-1}[r][u']}{\displaystyle\sum_{\forall u' \in V':(u',v') \in E'} w'(u')};$$

$$x^{**}[r][v'] = \omega \cdot x^*[r][v'] + (1 - \omega)x^{i-1}[r][u'];$$

        end
        for $v \in V'$ do
            // Rescale
            $l = \min_{u \in V'} x[r][u]$;
            $r = \max_{u \in V'} x[r][u]$;
            $x^i[r][v'] = \frac{x^{**}[r][v']-l}{r-l} - 0.5$;
        end
    end
end
for h in E do
    // Compute algebraic weights for hyperedges
    $alg\_weight(h) = 1 / \max_{\forall r} \max_{u,v \in h} |x[r][v] - x[r][u]|$;
end

4.2 Convergence analysis

algorithm 1 is an iterative process that computes $x[r][v']$ for all vertices $v' \in V'$ and random initial vector numbers r. To analyze the convergence of this process and understand the properties of the

computed algebraic weights, we need additional notation. Let $x^{(i)} \in \mathbb{R}^{|V'|}$ denote the ith iterate of the vector $x[r]$ for the rth random initial vector. Since the same iterative process is performed for all vectors of random initial coordinates, we will perform the analysis on only one (arbitrary) vector and hence omit the dependence on r.

Let $A \in \mathbb{R}^{|E| \times |V|}$ be the incidence matrix of the hypergraph; that is, $A_{ij} = 1$ if vertex j belongs to the hyperedge i and 0 otherwise. Let $S^v \in \mathbb{R}^{|V| \times |V|}$ and $S^h \in \mathbb{R}^{|E| \times |E|}$ be diagonal matrices such that

$$S_{jj}^v = w(v_j) \quad \text{and} \quad S_{ii}^h = \frac{w(h_i)}{|h_i|},$$

where $|h_i|$ denotes the cardinality of the ith hyperedge (same as in eq. (1)). Define the following matrix

$$W = \begin{bmatrix} 0 & A^T S^h \\ A S^v & 0 \end{bmatrix}$$

and let D be the diagonal matrix with elements $D_{jj} = \sum_i W_{ij}$. For convenience of analysis, we decompose D as

$$D = \begin{bmatrix} D^v & 0 \\ 0 & D^h \end{bmatrix}$$

with $D^v \in \mathbb{R}^{|V| \times |V|}$ and $D^h \in \mathbb{R}^{|E| \times |E|}$, such that the block sizes of D are compatible with those of W. Note that W is asymmetric.

It is not hard to see that algorithm 1 computes the following update:

$$x^{(i)} = \frac{1}{r-l} \Big[\underbrace{\omega D^{-1} W x^{(i-1)} + (1-\omega) x^{(i-1)}}_{x^{*(i-1)}} \Big] - \frac{r+l}{2(r-l)} \mathbf{1}$$

where $\mathbf{1}$ is the vector of all ones, and r and l are the maximum and the minimum of the elements in $x^{*(i-1)}$, respectively (see algorithm 1). Then, we simplify the update formula as

$$x^{(i)} = \alpha^{(i-1)} H x^{(i-1)} + \beta^{(i-1)} \mathbf{1}, \tag{3}$$

where

$$H = \omega D^{-1} W + (1-\omega) I, \quad \alpha^{(i-1)} = \frac{1}{r-l}, \quad \text{and} \quad \beta^{(i-1)} = -\frac{r+l}{2(r-l)}.$$

The convergence of the iteration depends on the properties of the iteration matrix H. We first note that the matrix $D^{-1} W$ is diagonalizable with real eigenvalues.

Theorem 1. *Let $(D^h)^{-1/2} (S^h)^{1/2} A (D^v)^{-1/2} (S^v)^{1/2}$ admit the following singular value decomposition:*

$$(D^h)^{-1/2} (S^h)^{1/2} A (D^v)^{-1/2} (S^v)^{1/2} = \sum_{i=1}^{r} \sigma_i u_i z_i^T.$$

Then, $D^{-1} W$ has a zero eigenvalue of multiplicity $|V| + |E| - 2r = |V'| - 2r$. Moreover, for the nonzero eigenvalues,

$$D^{-1} W \begin{bmatrix} (S^v)^{-1/2} (D^v)^{-1/2} z_i \\ (S^h)^{-1/2} (D^h)^{-1/2} u_i \end{bmatrix} = \sigma_i \begin{bmatrix} (S^v)^{-1/2} (D^v)^{-1/2} z_i \\ (S^h)^{-1/2} (D^h)^{-1/2} u_i \end{bmatrix} \tag{4}$$

and

$$D^{-1} W \begin{bmatrix} -(S^v)^{-1/2} (D^v)^{-1/2} z_i \\ (S^h)^{-1/2} (D^h)^{-1/2} u_i \end{bmatrix} = -\sigma_i \begin{bmatrix} -(S^v)^{-1/2} (D^v)^{-1/2} z_i \\ (S^h)^{-1/2} (D^h)^{-1/2} u_i \end{bmatrix}, \quad i = 1, \dots r. \tag{5}$$

10

Proof. The identities eq. (4) and eq. (5) are straightforward to verify based on the singular value decomposition. On the other hand, if λ is an eigenvalue of $D^{-1}W$ with corresponding eigenvector $\begin{bmatrix} c \\ d \end{bmatrix}$, then,

$$(D^v)^{-1}A^T S^h d = \lambda c \quad \text{and} \quad (D^h)^{-1}AS^v c = \lambda d,$$

which implies that

$$[(D^h)^{-1/2}(S^h)^{1/2}A(D^v)^{-1/2}(S^v)^{1/2}]^T[(D^h)^{1/2}(S^h)^{1/2}d] = \lambda[(D^v)^{1/2}(S^v)^{1/2}c],$$

and

$$[(D^h)^{-1/2}(S^h)^{1/2}A(D^v)^{-1/2}(S^v)^{1/2}][(D^v)^{1/2}(S^v)^{1/2}c] = \lambda[(D^h)^{1/2}(S^h)^{1/2}d].$$

In form, these two equalities define all nonzero singular values $|\lambda|$ of $(D^h)^{-1/2}(S^h)^{1/2}A(D^v)^{-1/2}(S^v)^{1/2}$. Thus, if λ is not equal to any of the $\pm\sigma_i$'s, then λ must be zero. $\qquad\square$

Second, note that the diagonal of the matrix $D^{-1}W$ is zero and each row is nonnegative, summing to one. Hence, by Gershgorin's circle theorem, the eigenvalues of $D^{-1}W$ must lie between -1 and 1. Indeed, the two ends of this interval are attainable.

Theorem 2. *The spectral radius of $D^{-1}W$ is 1. In particular, there exists a pair of eigenvalues ± 1, and $\mathbf{1}$ is an eigenvector associated with an eigenvalue $+1$.*

Proof. By the definition of W and D, we have $D^{-1}W\mathbf{1} = \mathbf{1}$. Therefore, $D^{-1}W$ has an eigenvalue $+1$ with eigenvector $\mathbf{1}$. From theorem 1 we know that $D^{-1}W$ must also have an eigenvalue -1. Then, because all the eigenvalues of $D^{-1}W$ lie between -1 and $+1$ by the Gershgorin's circle theorem, its spectral radius is 1. $\qquad\square$

Third, recall that the star expansion graph is a bipartite graph, with one part containing the vertices of the original hypergraph and the other part the hyperedges. Rather than defining vertex weights as in eq. (1) and leaving the edges unweighted, alternatively, one may view W as the weighted adjacency matrix of a graph with exactly the same vertex-edge structure, but the edges are weighted according to W and the vertices are left unweighted. Then, the block and the sparsity structure of W indicates that the directed edges of this bipartite graph (in the alternative view) always come in pairs with opposite directions. Hence, we may ignore the directions and treat the graph undirected, if at all convenient for analyzing graph connectivity. Then, the directed version of the graph is strongly connected if and only if the undirected version is connected, although the edge weights are asymmetric in the directed version. Consequently, the number of strongly connected components of the directed version is equal to the number of connected components of the undirected version. Such a view allows the application of the Perron–Frobenius theorem for exploring the extreme eigenvalues of $D^{-1}W$.

Theorem 3. *The eigenvalues of $D^{-1}W$ equal to $+1$ are all simple. The number of such eigenvalues is equal to the number of strongly connected components of the directed version of the bipartite graph (or equivalently, the number of connected components of the undirected version of the graph).*

Proof. If the directed graph is strongly connected, then $D^{-1}W$ is irreducible. Hence, by the Perron–Frobenius theorem, there exists an eigenvalue equal to the spectral radius 1. This eigenvalue is simple and is unique. If the graph has C strongly connected components, then $D^{-1}W$ may be

11

symmetrically permuted into a $C \times C$ block-diagonal matrix, where each block corresponds to one strongly connected component. In such a case, $D^{-1}W$ has C eigenvalues equal to 1, all of which are simple. □

The above three theorems reveal the beautiful symmetry of the eigenvalues of $D^{-1}W$. They are real, come in pairs, and straddle around zero (except those being exactly zero). The number of nonzero pairs is equal to the rank of the incidence matrix A. The eigenvalues all lie inside $[-1, 1]$. Moreover, there exists (at least) one pair attaining exactly ± 1, and the number of such pairs is equal to the number of connected components of the star expansion graph when viewed as undirected.

With this knowledge, we see that there is a one-to-one correspondence between the eigenpairs of the iteration matrix H and those of $D^{-1}W$. Specifically, denote by (μ_i, ϕ_i) an eigenpair of H. Then,

$$H\phi_i = \mu_i\phi_i \quad \Leftrightarrow \quad D^{-1}W\phi_i = \frac{\mu_i - 1 + \omega}{\omega}\phi_i, \quad 0 < \omega < 1.$$

Because the eigenvalues of $D^{-1}W$ are symmetric around 0 with range $[-1, 1]$, the eigenvalues μ_i of H are symmetric around $1 - \omega$ with range $[1 - 2\omega, 1]$. Because ω is strictly less than 1, the largest eigenvalue of H is 1, which is equal to the largest eigenvalue in magnitude. The multiplicity of this eigenvalue is equal to the number of connected components of the undirected version of the star expansion graph. For simplicity of analysis, we will assume from now on that the graph is connected. Then, we have an ordering of the eigenvalues according to their magnitude:

$$1 = \mu_1 > |\mu_2| \geq |\mu_3| \geq \cdots \geq |\mu_{|V'|}|,$$

with $\phi_1 = \mathbf{1}$ being an (unnormalized) eigenvector associated with μ_1. Note the use of the strictly-greater-than sign $>$ and the greater-than-or-equal-to sign $\geq$. In particular, the second eigenvalue, in magnitude, must be strictly less than 1. In what follows, we will present a result that relates the difference between the elements of $x^{(k)}$ to that between the corresponding elements of some vector in the eigensubspace spanned by one or a few eigenvectors, including ϕ_2. This eigensubspace depends on how many eigenvalues are equal to $|\mu_2|$ in magnitude. If only one, that is, $|\mu_2| > |\mu_3|$, then the eigensubspace is spanned by only ϕ_2. However, if more than one, then let us assume that $|\mu_2| = |\mu_3| = \cdots = |\mu_t| > |\mu_{t+1}|$. Such a case includes two subcases:

case 1: $\mu_2 = \mu_3 = \cdots = \mu_t$; and

case 2: $\mu_2, \mu_3, \ldots, \mu_t$ are not all equal.

In each subcase, the vector is some linear combination of $\phi_2, \ldots, \phi_t$.

Theorem 4. *Assume that the undirected version of the star expansion graph is connected. Let the initial iterate $x^{(0)}$ be expanded in the eigen-basis of H as*

$$x^{(0)} = a_1\phi_1 + a_2\phi_2 + \cdots + a_{|V'|}\phi_{|V'|}.$$

(i) If $\mu_2 = \mu_3 = \cdots = \mu_t$ and $|\mu_t| > |\mu_{t+1}|$ for some $t \geq 2$, and if $a_2, \ldots, a_t$ are not all zero, then for any pair i, j,

$$\lim_{k \to \infty} \frac{(x^{(k)})_i - (x^{(k)})_j}{\alpha^{(0)}\alpha^{(1)}\cdots\alpha^{(k-1)}\mu_2^k} = \xi_i - \xi_j,$$

where

$$\xi = a_2\phi_2 + \cdots + a_t\phi_t.$$

12

(ii) *If $|\mu_2| = |\mu_3| = \cdots = |\mu_t| > |\mu_{t+1}|$ for some $t \geq 3$ where $\mu_2, \mu_3, \ldots, \mu_t$ are not all equal, and if $a_2, \ldots, a_t$ are not all zero, then for any pair i, j,*

$$\lim_{k \to \infty} \frac{(x^{(2k+p)})_i - (x^{(2k+p)})_j}{\alpha^{(0)} \alpha^{(1)} \cdots \alpha^{(2k+p-1)} \mu_2^{2k+p}} = (\eta_p)_i - (\eta_p)_j,$$

where

$$\eta_p = a_2 \phi_2 + a_3 (\mu_3/\mu_2)^p \phi_3 + \cdots + a_t (\mu_t/\mu_2)^p \phi_t, \quad p = 0, 1.$$

Proof. For notational convenience, let $n = |V'|$. Write the diagonalization of H as $H = \Phi \Lambda \Phi^{-1}$, where $\Phi = [\phi_1, \ldots, \phi_n]$ and $\Lambda = \text{diag}(\mu_1, \ldots, \mu_n)$. Also write $x^{(0)} = \Phi a$, where $a = [a_1, \ldots, a_n]^T$. Then, expanding (3) k times, we have

$$\begin{aligned}
x^{(k)} &= \alpha^{(0)} \cdots \alpha^{(k-1)} H^k x^{(0)} + \\
&\quad + (\beta^{(0)} \alpha^{(1)} \cdots \alpha^{(k-1)} H^{k-1} + \beta^{(1)} \alpha^{(2)} \cdots \alpha^{(k-1)} H^{k-2} + \cdots + \beta^{(k-1)} H^0) \mathbf{1}.
\end{aligned} \tag{6}$$

Because $\mathbf{1}$ is an eigenvector of H corresponding to eigenvalue 1, the term in the second line is equal to $\gamma^{(k)} \mathbf{1}$ with $\gamma^{(k)} = \beta^{(0)} \alpha^{(1)} \cdots \alpha^{(k-1)} + \beta^{(1)} \alpha^{(2)} \cdots \alpha^{(k-1)} + \cdots + \beta^{(k-1)}$. For the first term on the right of the first line,

$$H^k x^{(0)} = \Phi \Lambda^k a = a_1 \phi_1 + a_2 \mu_2^k \phi_2 + \cdots + a_n \mu_n^k \phi_n,$$

with $\phi_1 = \mathbf{1}$. Therefore,

$$x^{(k)} = \alpha^{(0)} \cdots \alpha^{(k-1)} (a_1 \mathbf{1} + a_2 \mu_2^k \phi_2 + \cdots + a_n \mu_n^k \phi_n) + \gamma^{(k)} \mathbf{1},$$

and hence

$$\frac{(x^{(k)})_i - (x^{(k)})_j}{\alpha^{(0)} \cdots \alpha^{(k-1)} \mu_2^k} = (e_i - e_j)^T [a_2 \phi_2 + a_3 (\mu_3/\mu_2)^k \phi_3 + \cdots + a_n (\mu_n/\mu_2)^k \phi_n].$$

(i) When k is large, the term inside the square bracket is dominated by

$$a_2 \phi_2 + \cdots + a_t \phi_t,$$

because all other μ_i's are smaller than μ_2 in magnitude. Thus, when $k \to \infty$, only this term remains, hence the result.

(ii) Similar to the above case,

$$\frac{(x^{(2k+p)})_i - (x^{(2k+p)})_j}{\alpha^{(0)} \cdots \alpha^{(2k+p-1)} \mu_2^{2k+p}} = (e_i - e_j)^T [a_2 \phi_2 + a_3 (\mu_3/\mu_2)^{2k+p} \phi_3 + \cdots$$

$$\cdots + a_n (\mu_n/\mu_2)^{2k+p} \phi_n].$$

When $k \to \infty$, in the square bracket only the term

$$a_2 \phi_2 + a_3 (\mu_3/\mu_2)^{2k+p} \phi_3 + \cdots + a_t (\mu_t/\mu_2)^{2k+p} \phi_t$$

remains and the rest vanishes. Because the eigenvalues $\mu_2, \ldots, \mu_t$ are all real, the ratios $\mu_3/\mu_2, \ldots \mu_t/\mu_2$ can only be ± 1. Hence, taking square, the ratios all become 1. We thus obtain the result in the theorem.

13

□

Informally speaking, the above theorem states that in the limit, the difference between two elements of the iterate vector $x^{(k)}$ is proportional to that between the corresponding elements of some vector ξ. When $|\mu_2|$ is strictly greater than $|\mu_3|$, this vector ξ may be taken to be the eigenvector ϕ_2. When there exist more than one eigenvalue equal to μ_2, say, $\mu_2 = \cdots = \mu_t$ for some $t > 2$, then ξ is a linear combination of $\phi_2, \ldots, \phi_t$, where the coefficients of the combination are the expansion coefficients of the initial iterate $x^{(0)}$ along the eigen-basis of the iteration matrix H. However, when there exist more than one eigenvalue whose magnitude is equal to $|\mu_2|$ but these eigenvalues are not all the same, then the situation is slightly complicated. Every other iterate $x^{(k)}$ form a subsequence and the limiting behaviors of these two interleaving subsequences correspond to two different vectors, which are η_0 and η_1 defined in the theorem. Both of them are a linear combination of the eigenvectors $\phi_2, \ldots, \phi_t$. For η_0, the coefficients of the combination are the expansion coefficients of $x^{(0)}$ along the eigen-basis of H; for η_1, some of these coefficients flip signs.

The existence of these various cases is owing to the different choices of ω. Recall that the eigenvalues of $D^{-1}W$ are symmetrically distributed around zero. Then, the eigenvalues of $H = \omega D^{-1}W + (1 - \omega)I$ are symmetrically distributed around $1 - \omega$, with the smallest one being $1 - 2\omega$. We will use $\sigma_2 > 0$ to denote the second largest eigenvalue of $D^{-1}W$, which is consistent with the notation in Theorem 1. By the connected-graph assumption, $\sigma_2 < \sigma_1 = 1$. We also have that the multiplicity of $1 - 2\omega$ is one, by the same assumption. Furthermore, the second largest eigenvalue of H is $\omega\sigma_2 + 1 - \omega$.

(a) If $\omega > \frac{2}{3 - \sigma_2}$, then $1 - 2\omega$ is negative and it has a larger magnitude than does $\omega\sigma_2 + 1 - \omega$. In such a case, $\mu_2 = 1 - 2\omega$ and $|\mu_2|$ is strictly greater than $|\mu_3|$. This scenario corresponds to the case (i) of Theorem 4 with $t = 2$.

(b) If $\omega \leq \frac{1}{2}$, then $1 - 2\omega$ is nonnegative, and hence all eigenvalues of H are nonnegative. In such a case, $\mu_2 = \omega\sigma_2 + 1 - \omega$. This scenario also corresponds to the case (i) of Theorem 4, because if there are more than one eigenvalue whose magnitude is equal to μ_2, then the nonnegativity implies that these eigenvalues must be the same as μ_2. The multiplicity of μ_2 is equal to the multiplicity of σ_2. It could happen that either the multiplicity of σ_2 is one, in which case $t = 2$; or the multiplicity of σ_2 is greater than one, in which case $t > 2$.

(c) If $\omega = \frac{2}{3 - \sigma_2}$, then $1 - 2\omega$ is negative, $\omega\sigma_2 + 1 - \omega$ is positive, and the two have the same magnitude. This scenario corresponds to the case (ii) of theorem 4. Whether $t = 3$ or $t > 3$ depends on the multiplicity of $\omega\sigma_2 + 1 - \omega$ (equivalently that of σ_2), because the multiplicity of $1 - 2\omega$ is one. If the multiplicity of σ_2 is one, then only two eigenvalues have magnitude equal to $|\mu_2|$, and hence $t = 3$; otherwise, $t > 3$.

(d) If $\frac{1}{2} < \omega < \frac{2}{3 - \sigma_2}$, then $1 - 2\omega$ is negative and $\omega\sigma_2 + 1 - \omega$ is greater than the magnitude of $1 - 2\omega$. This scenario is similar to that of the above item (b), except that not all the eigenvalues of H are nonnegative. All other properties are otherwise the same.

4.3 Mutually influenced model

In the standard graph case, the edge weights provide a first-order measurement of vertex similarity/distance. This measure is applicable to only adjacent vertices. Hence, for a pair of

vertices that are not adjacent, global information of the graph must be incorporated for extending the measurement. The *algebraic distance on graphs* [15] defines algebraic coordinates for every vertex through an iterative process similar to that in the present work. In the limit, the coordinate difference, which serves the notion of "distance," is proportional to the difference of the corresponding elements of some vector y. Specifically, if we let w_{ij} be the weights of a pair of vertices ij in the graph, then the elements of the vector y satisfy an equilibrium state

$$y_i = \gamma y_i + \sum_j \frac{w_{ij}}{d_i} y_j, \tag{7}$$

where $d_i = \sum_j w_{ij}$ is used for normalization and $0 < \gamma < 1$ is a constant factor for all i. The algebraic coordinates do not need to coincide with the y_i's; it suffices for their differences to be proportional. When one treats the vertices to be entities of a mutually influenced environment, then the value y_i of each entity is composed of two components according to (7): a portion of itself (γy_i) and a normalized weighted contributions of its neighbors ($\sum_j \frac{w_{ij}}{d_i} y_j$). If y_i denotes the amount of information stored at vertex i, eq. (7) essentially signifies a global equilibrium of the flow of information. Such a view is used to interpret the global similarity of vertices through neighborhoods. In short, two vertices are similar if their neighborhoods are similar, because the common factor γ is constant and similarity relies on the neighboring weights w_{ij}.

The present work extends this notion to hypergraphs. In such a setting, the hyperedges cover not just a pair ij of vertices, but rather, a subset of any size (excluding of course empty sets and singletons). Hence, a proxy of the pairwise environment is the star expansion graph G, whose vertex set V' includes not only the original vertices V of the hypergraph, but also the hyperedges E. Naturally, the edges of G come from the containment relation between V and E; that is, $v \in V$ and $h \in E$ are connected if and only if $v \in h$. Therefore, the edge weights of G come from the weights in the original hypergraph, properly scaled. It is important to note that the scaling is not symmetric, because potentially one hyperedge may contain a large number of vertices. Thus, whereas the weights for the directed edge from v to h are $w(v)$ without scaling, those for the edges from h to v are $w(h)$ scaled by $|h|$.

Then, the *algebraic distance on hypergraphs* enjoys the same equilibrium model eq. (7). From theorem 4 and the subsequent discussions, we know that $\gamma = (1 - \mu_2)/\omega$, where μ_2 is the second largest eigenvalue in magnitude of the iteration matrix H, and y is a vector in the subspace spanned by the eigenvectors associated with eigenvalues whose magnitude are equal to $|\mu_2|$. In practice, ω is often set to be $1/2$, then μ_2 is positive[1] and all other eigenvalues having the same magnitude must be equal to μ_2. If furthermore the multiplicity of μ_2 is 1, then the subspace is spanned by the eigenvector ϕ_2 only and hence y can be taken to be ϕ_2. Note that the multiplicity of eigenvalue μ_2 is the same as the multiplicity of the singular value σ_2 of the matrix $(D^h)^{-1/2}(S^h)^{1/2}A(D^v)^{-1/2}(S^v)^{1/2}$ in theorem 1.

Finally, when extending the notion of the *algebraic distance* to hypergraphs, we have to take into account the non-pairwise nature of the relations between the vertices. While in pairwise graph setting it is natural to use the simple difference between two vertices' algebraic coordinates as the measure of similarity between them, in non-pairwise hypergraph setting this approach has to be extended. In the hypergraph case we instead assume that the hyperedge is only as important as two most dissimilar vertices in it. In hypergraph partitioning terms, this means that we want to

---

[1]We assume the nondegenerate case where μ_2 is nonzero. Otherwise, all eigenvalues except for the largest one is zero.

avoid cutting the hyperedges that contain only similar vertices. Therefore we define the algebraic weight of the hyperedge as:

$$alg\_weight(h) = \frac{1}{\max\{|x[v] - x[u]| \mid u, v \in h\}}$$

This way, we penalize cutting the hyperedges that contain only similar vertices, with the notion of similarity defined according to the mutually influenced model.

5   Experimental Results

In this section, we first illustrate the empirical convergence of algorithm 1 and then compare the quality of the partitioning produced by our algorithm with those by other state-of-the-art hypergraph partitioners. Our major goal is to study the effectiveness of the algebraic distance on hypergraph partitioning by introducing it into a coarsening scheme. We note that these results can clearly be further improved by using more advanced refinement techniques which are beyond the scope of this work.

5.1   Convergence

The speed of the convergence of algebraic weights depends on the gap between the second largest eigenvalue in magnitude, $|\mu_2|$, of the iteration matrix H and the next eigenvalue with a different magnitude, denoted as $|\mu_{t+1}|$ in the preceding section. Estimating this gap is no less expensive than computing the corresponding eigenvectors. Hence, in practice, we use the squared sine of the angle between two iterates $x^{(k)}$ and $x^{(k+1)}$:

$$1 - \left\langle \frac{x^{(k)}}{||x^{(k)}||}, \frac{x^{(k+1)}}{||x^{(k+1)}||} \right\rangle^2,$$

to measure how parallel the two iterates are, as a proxy of convergence test.

The two consecutive iterates generally become parallel very quickly. In fig. 3 we pick five hypergraphs and plot the squared sine for the first 50 iterations. These hypergraphs represent different sizes (from $2426 - by - 3602$ to $103631 - by - 395979$) and different origins (from social networks to circuit simulation) of hypergraphs in the benchmark. One sees that the value is indistinguishable from zero after 10 to 20 iterations. Such a phenomenon is typical to our experience and we generally set num_iter comparable to these numbers.

It is worth noting that the parallelism of two consecutive iterates does not necessarily mean true convergence. As noted by Chen and Safro [15], the eigenvalue gap may be so small that it might take a huge number of iterations for the difference

$$(x^{(k)})_i - (x^{(k)})_j$$

(after scaling) to get close enough to $\xi_i - \xi_j$. The parallelism before convergence corresponds to a transient state wherein the change of the iterates is small. It turns out that in practice, a transient state is sufficient for the algebraic distance to be useful for coarsening.
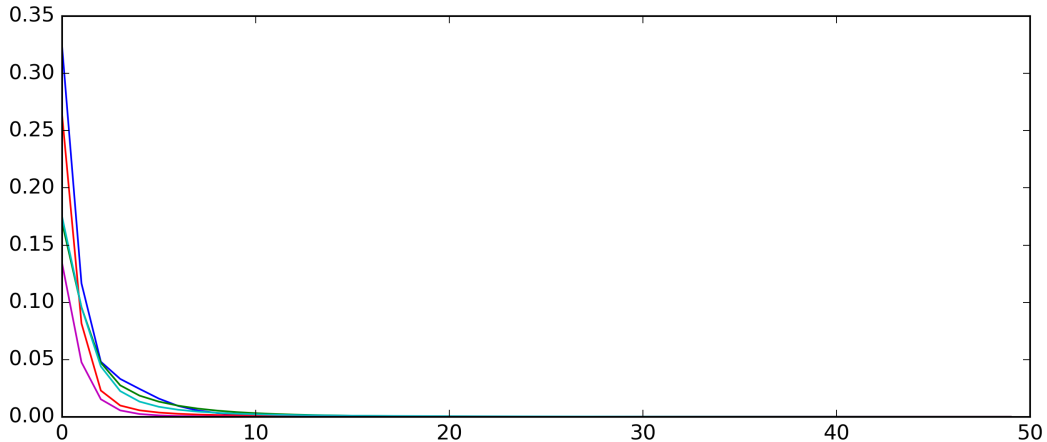
Figure 3: Squared sine of the angle between $x^{(k)}$ and $x^{(k+1)}$ as a function of the iteration number k.

5.2 Cut improvements

We implement our algorithm with algebraic distances by using the Zoltan [18] package of the Trilinos Project [27]. Zoltan is an open-source toolkit of parallel combinatorial scientific computing algorithms [18] designed to optimize load balancing. It includes a hypergraph partitioning algorithm called PHG (Parallel HyperGraph partitioner). We augment Zoltan's PHG partitioner with algebraic distances as described in the preceding section. Our algorithm is called Zoltan-AlgD. All comparisons with Zoltan are indeed comparisons with Zoltan's PHG partitioner. In this study we use Zoltan in the serial mode. In addition to comparing Zoltan-AlgD to Zoltan, we compare it with two other state-of-the-art partitioners, namely hMetis2 [33] and PaToH v3.2 [11].

PaToH is used as a plug-in for Zoltan, as described in Zoltan's User Guide [19]. We run PaToH with two different parameters: default (here denoted as PaToH-D) and quality (PaToH-Q). hMetis2 is used to directly optimize k-way partitioning. All parameters are set as default: greedy first-choice scheme for coarsening, random k-way refinement, and min-cut objective function.

Each algorithm is run 10 times and the smallest cuts over all runs are compared. In the experiments, standard deviation of the cuts is usually small ($< 5\%$). Interestingly, in a small number of hypergraphs, we observe that the distribution of cuts is bimodal (i.e., the partitioner produces cuts close to either one of two modes). In this case, the standard deviation is high; however, within each mode, the deviation is low. Such behavior demonstrates that in certain settings the solvers cannot escape false local attraction basins obtained at the coarse levels. This hints that the current state of hypergraph partitioning solvers is still far from being optimal and there is a lot of space for improvement.

Because the implementation of our algorithm has not been optimized, the run time of Zoltan-AlgD is on average two to four times longer than that of Zoltan (see fig. 8). However, since the iteration scheme is easily parallelizable (for example, among different random vectors, the iteration is pleasingly parallel as well as parallelization of Jacobi-based relaxations has been studied and used in many works), the overhead of computing the algebraic distances may be minimized and thus the run time may be made similar to that of Zoltan. Moreover, the algebraic distance is, in

fact, several iterations of Jacobi over-relaxation whose parallelization has been studied a lot.

We also have to point out the lack of progress in coarsening techniques in the recent years, with all major hypergraph partitioning packages using the same approach, making our contribution more valuable. While the way we select vertices to be merged together during coarsening is indeed more expensive than traditional approaches, it introduces very low overhead (*constant* number of passes over vertices) and shows great improvement in the area where state-of-the-art has not changed for a long time.

The different algorithms are compared on three groups of hypergraphs: big-bench, SNAP-bench, and social-networks-bench.

The first two groups (big-bench and SNAP-bench) are generated from matrices obtained from the University of Florida Sparse Matrix Collection [16] by using a row-net model: a vertex i belongs to hyperedge j if there is a non-zero element on the intersection of ith column and jth row. Big-bench contains 443 matrices; many of them are derived from optimization problems. SNAP-bench consists of 42 matrices from the SNAP (Stanford Network Analysis Platform) Network Data Sets [37].

The third group of the benchmark, social-networks-bench (12 hypergraphs), has two parts. The first part contains two networks with known communities (youtube and flickr) from the IMC 2007 Data Set [40]. The hypergraphs are constructed in the following way: each vertex represents a user and each hyperedge represents a community. After generating the hypergraph, isolated vertices are removed. The second part contains "similar hypergraphs." They are generated by using the following pipeline: a graph of pairwise links is obtained for the same dataset and then a similar graph is generated by using the BarabasiAlbertGenerator [3, 4] of NetworKit [50]. Afterwards, the adjacency matrix of the new graph is interpreted by using the row-net model.

In figs. 4 to 7 the results are presented graphically. Each curve plots the ratio

$$\frac{\text{cut obtained using another algorithm}}{\text{cut obtained using Zoltan-AlgD}},$$

where for the brown curve, the "other algorithm" is Zoltan; aquamarine curve, PaToH-Q; blue curve, PaToH-D; and orange curve, hMetis2. Each plot corresponds to a certain number of parts and a certain imbalance factor. The hypergraphs are ordered in the the increasing ratio.

For readability, the results for big-bench are split in two parts: those with ratios

$$\frac{\text{cut obtained using another algorithm}}{\text{cut obtained using Zoltan-AlgD}} < 1.5$$

and those with ratios greater than 1.5 (i.e., with more than 50% improvement in cut). The results for the social-networks-bench and SNAP-bench are plotted on the same figure, since the social-networks-bench is considerably smaller than the other two. The results show substantial improvement over Zoltan without algebraic distance, as well as over hMetis2 and PaToH on most of the hypergraphs. For the full set of results, please refer to `http://shaydul.in/hypergraph-partitioning-archive/`.

6 Conclusions

In this paper we have introduced a new similarity measure for hypergraph vertices—algebraic distances. This similarity measure is used for more accurate vertex aggregation during the coarsening
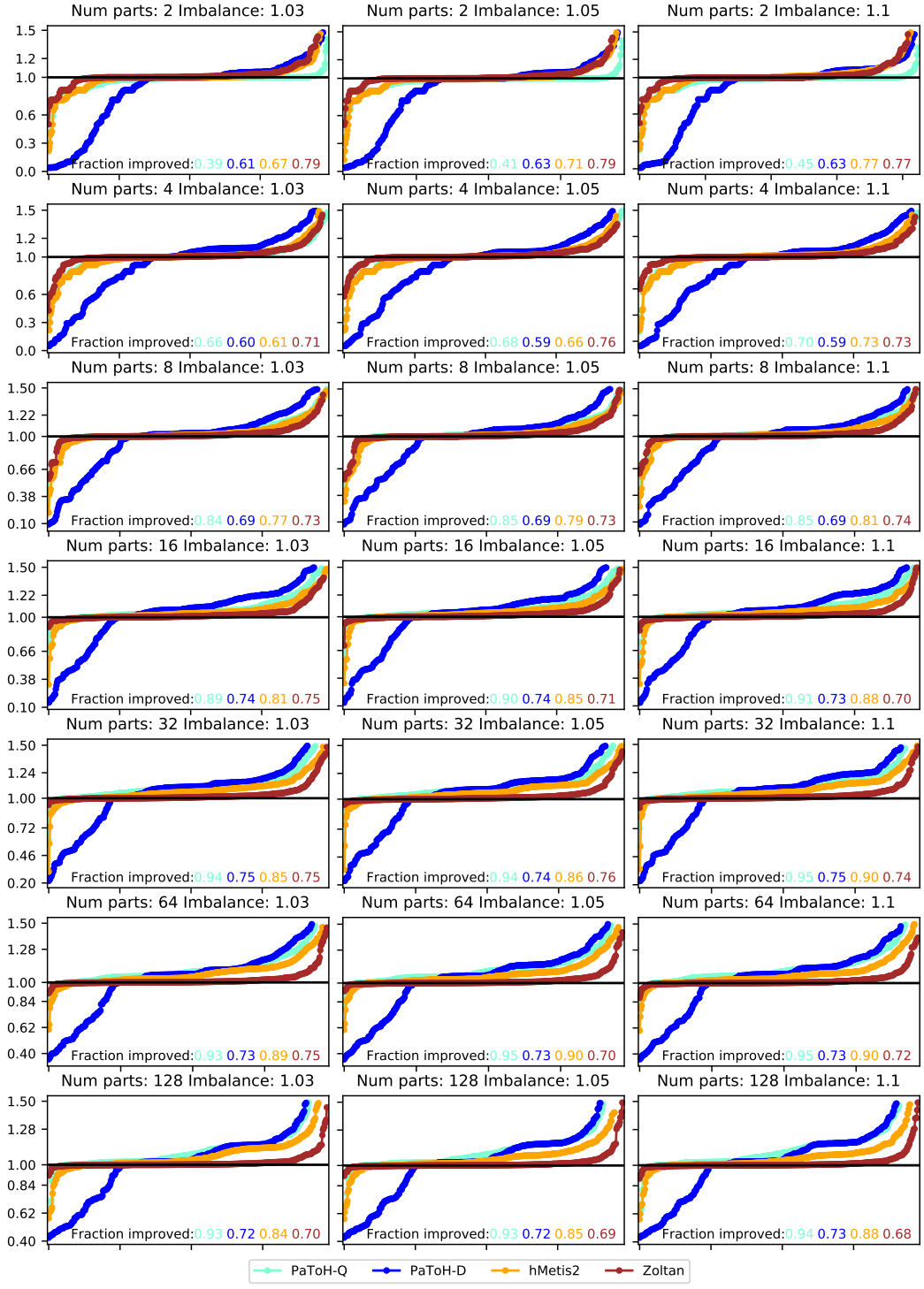
Figure 4: First half ($\frac{\text{cut obtained using another algorithm}}{\text{cut obtained using Zoltan-AlgD}} < 1.5$) of big-bench (443 hypergraphs), with brown curve corresponding to Zoltan, aquamarine to PaToH-Q, blue to PaToH-D, and orange to hMetis2. The improvements are with respect to the whole big-bench, including those with improvement greater than 1.5.
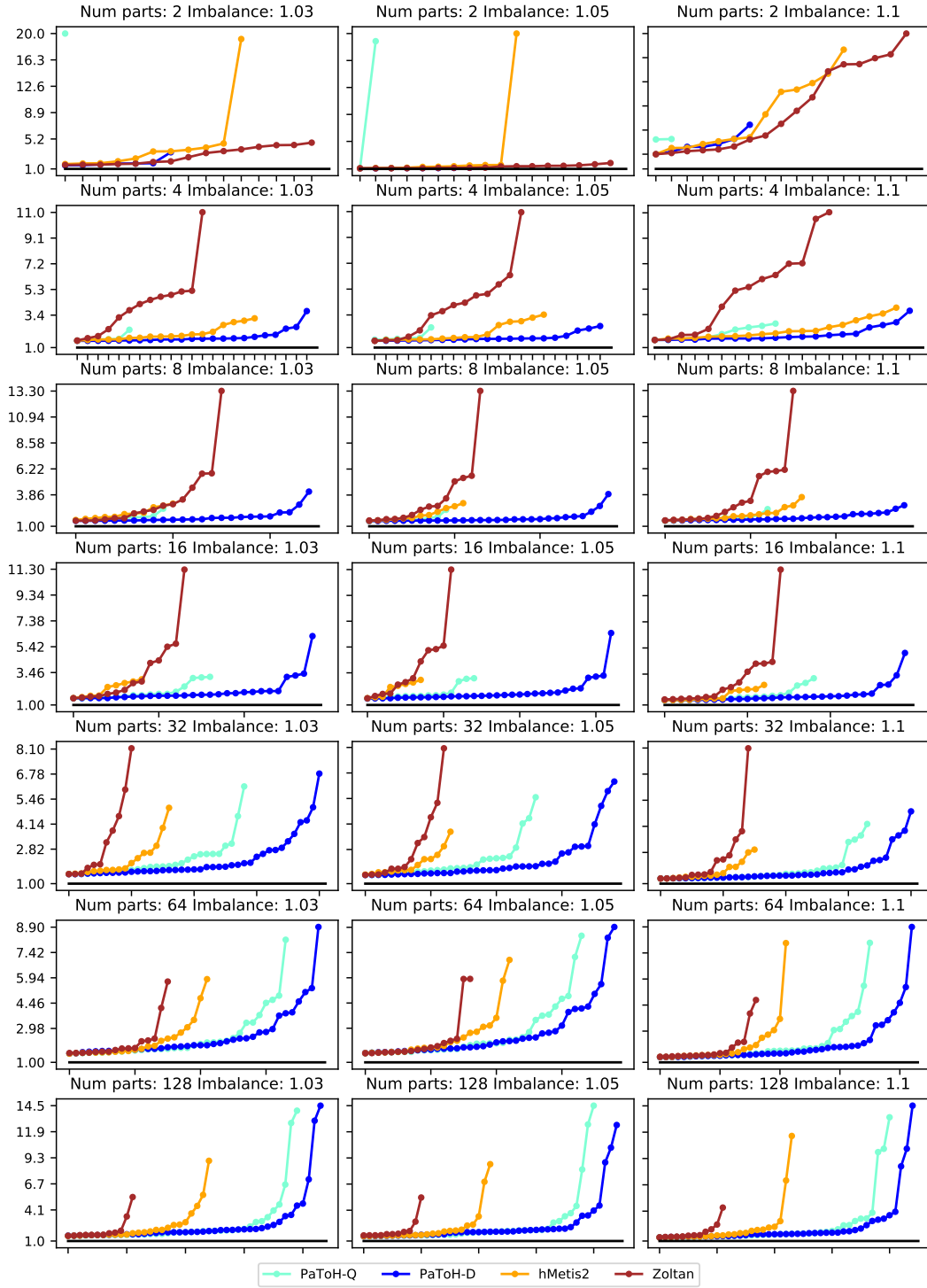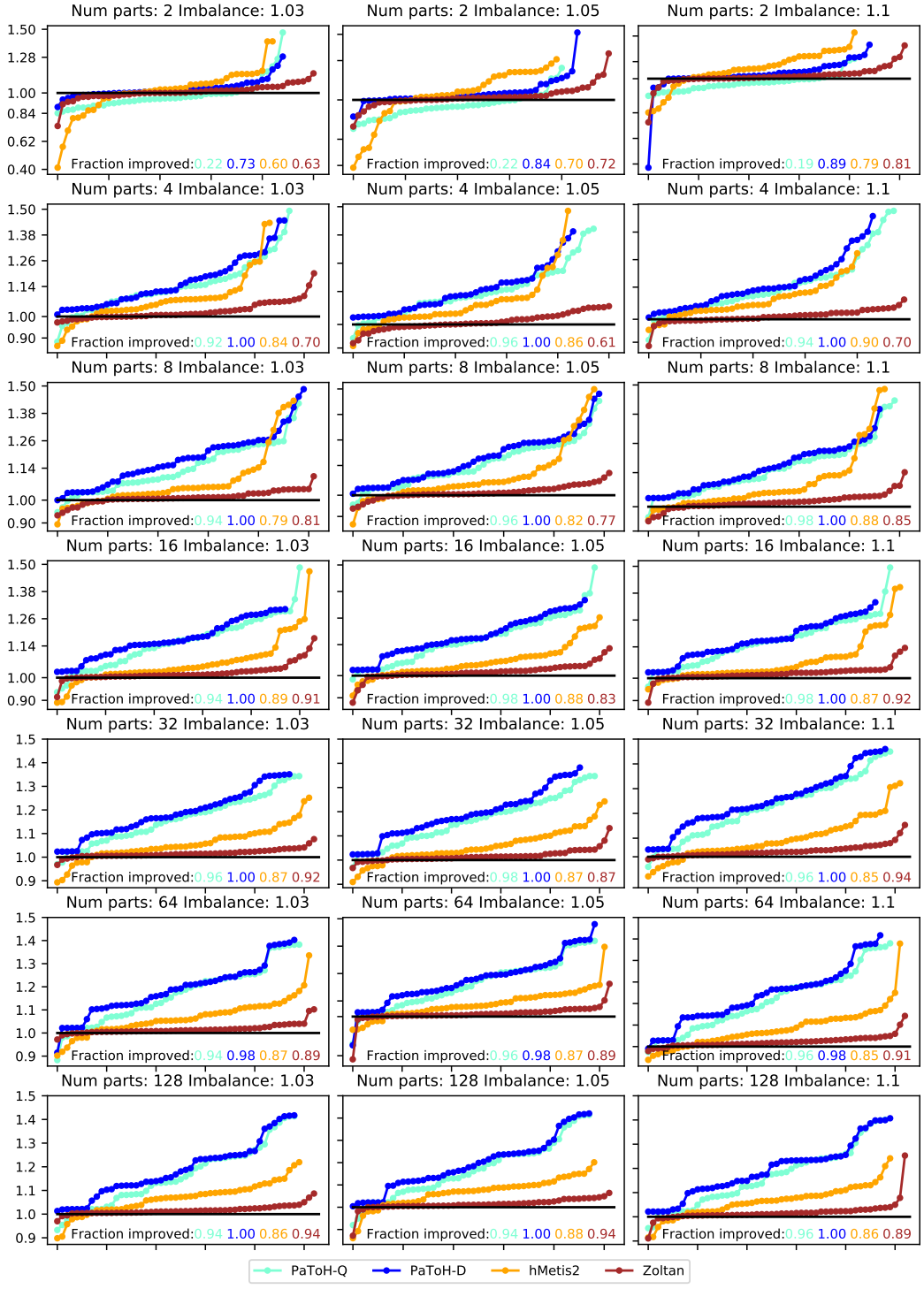
Figure 5: Second half ($\frac{\text{cut obtained using another algorithm}}{\text{cut obtained using Zoltan-AlgD}} \geq 1.5$) of big-bench (443 hypergraphs), with brown curve corresponding to Zoltan, aquamarine to PaToH-Q, blue to PaToH-D, and orange to hMetis2 (greater is better).

Figure 6: First half ($\frac{\text{cut obtained using another algorithm}}{\text{cut obtained using Zoltan-AlgD}} < 1.5$) of SNAP-bench and social-networks-bench (54 hypergraphs), with brown curve corresponding to Zoltan, aquamarine to PaToH-Q, blue to PaToH-D, and orange to hMetis2. The improvements are with respect to the whole big-bench, including those with improvement greater than 1.5
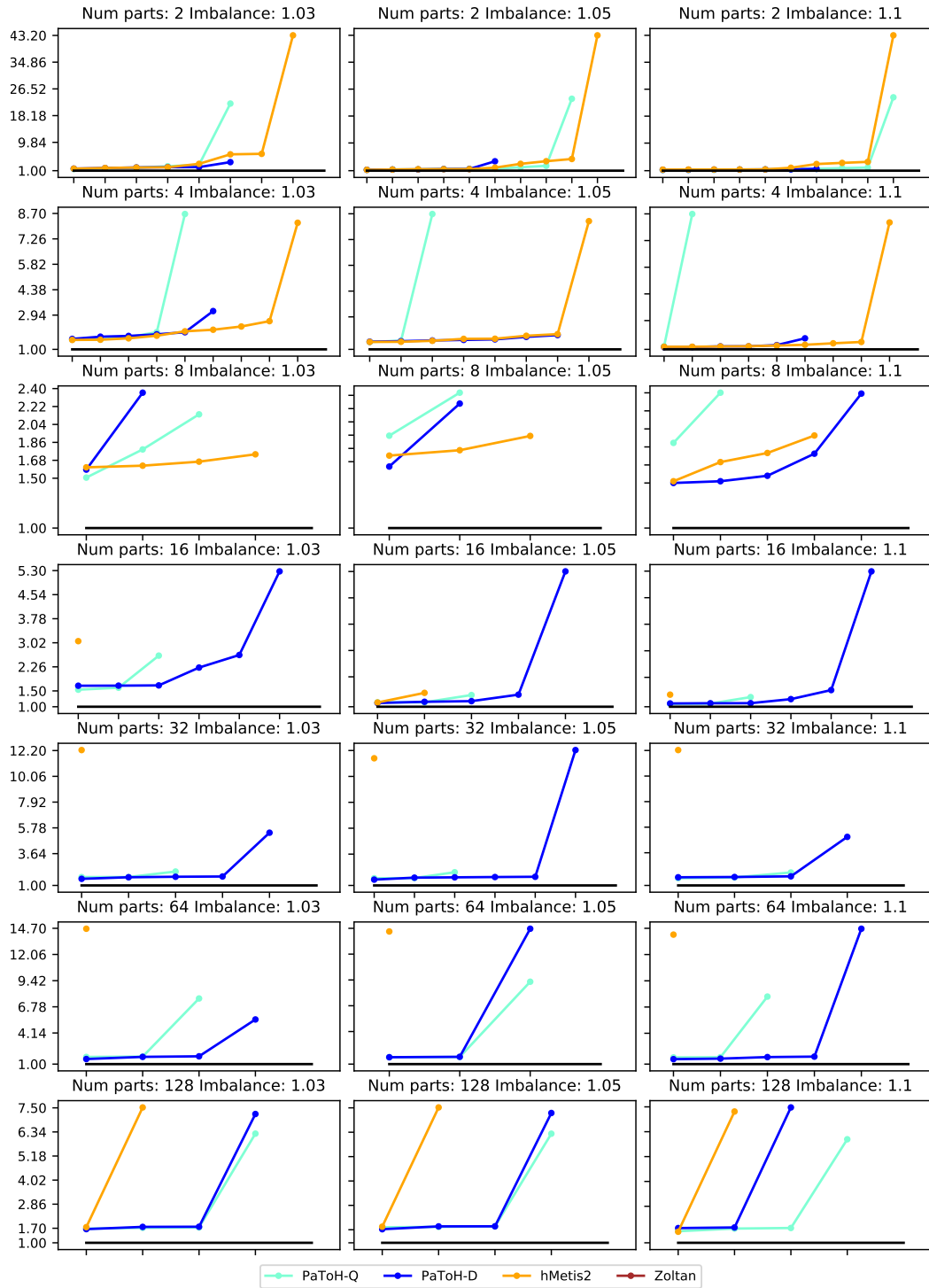
Figure 7: Second half ($\frac{\text{cut obtained using another algorithm}}{\text{cut obtained using Zoltan-AlgD}} \geq 1.5$) of SNAP-bench and social-networks-bench (54 hypergraphs), with brown curve corresponding to Zoltan, aquamarine to PaToH-Q, blue to PaToH-D, and orange to hMetis2 (greater is better).
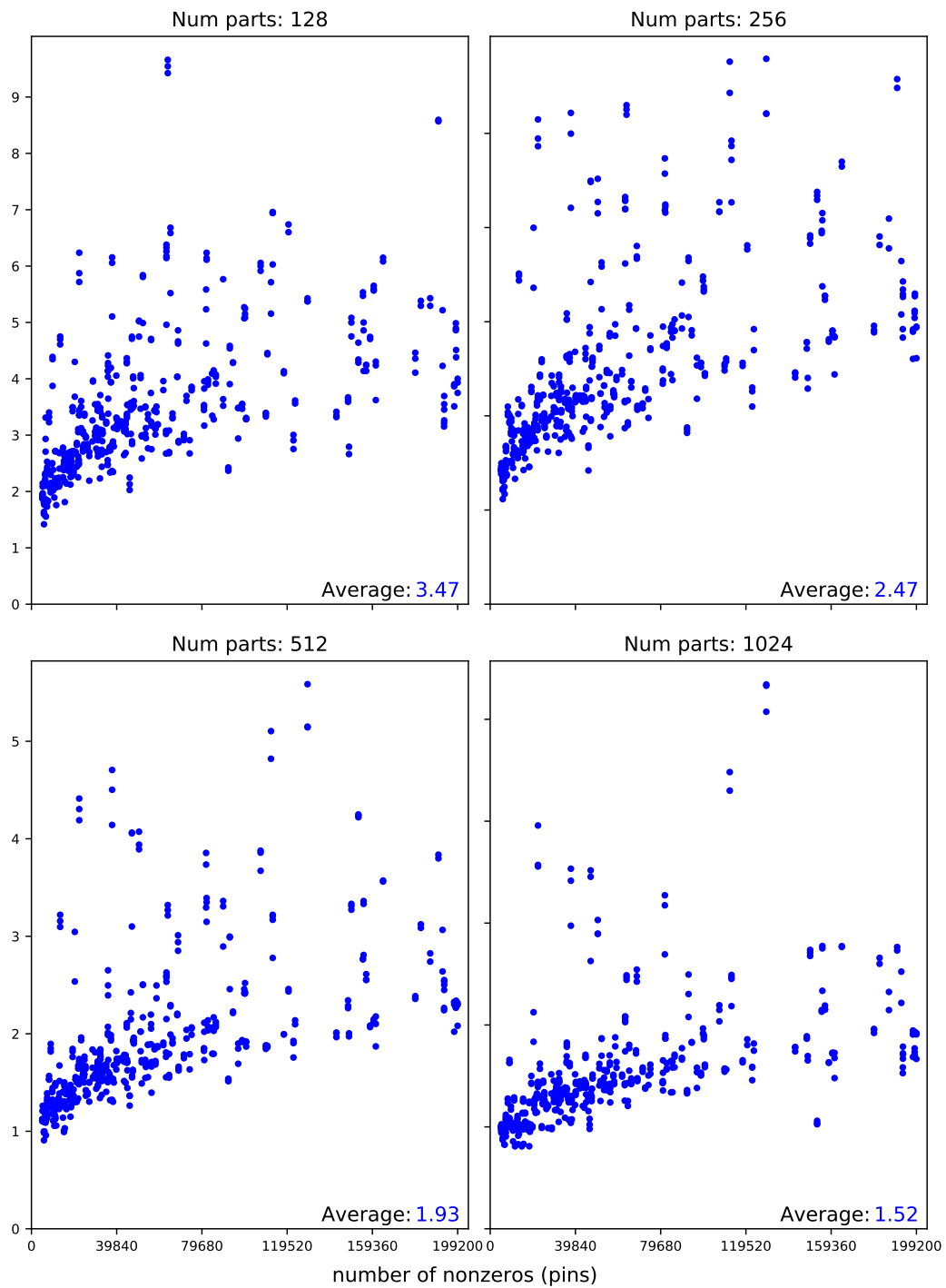
Figure 8: Ratio $\frac{\text{Zoltan-AlgD runtime}}{\text{Zoltan runtime}}$ for the big-bench benchmark. Time is CPU time as reported by *zdrive*.

stage of a multilevel algorithm. A serial iterative procedure for computing algebraic distances is introduced and implemented within the multilevel hypergraph partitioning framework Zoltan. The procedure results in a significant improvement (average of 34.3%) over the same framework without algebraic distances, while decreasing the cut by more than two times for some hypergraphs. The algorithm is shown to outperform other state-of-the-art partitioners as well.

The experimental results indicate that one may gain substantial performance improvements through exploiting the global structure of highly irregular hypergraphs (e.g. social networks and other hypergraphs with power-law degree distribution). Exploiting the spectral properties of the hypergraph and its star expansion through some iterative procedure, like the one proposed in the work, is one way to achieve the gain. There remains ample room for improvement for current state-of-the-art hypergraph partitioners, particularly for the coarsening stage.

Acknowledgments

References

[1] Y. Akhremtsev, T. Heuer, P. Sanders, and S. Schlag, *Engineering a direct k-way hypergraph partitioning algorithm*, in 2017 Proceedings of the Ninteenth Workshop on Algorithm Engineering and Experiments (ALENEX), SIAM, 2017, pp. 28–42.

[2] C. J. Alpert and A. B. Kahng, *Recent directions in netlist partitioning: a survey*, Integration, the VLSI journal, 19 (1995), pp. 1–81.

[3] S. T. Barnard and H. D. Simon, *Fast multilevel implementation of recursive spectral bisection for partitioning unstructured problems*, Concurrency and computation: Practice and Experience, 6 (1994), pp. 101–117.

[4] V. Batagelj and U. Brandes, *Efficient generation of large random networks*, Physical Review E, 71 (2005), p. 036113.

[5] C.-E. Bichot and P. Siarry, *Graph partitioning*, John Wiley & Sons, 2013.

[6] M. Bolla, *Spectra, euclidean representations and clusterings of hypergraphs*, Discrete Mathematics, 117 (1993), pp. 19–39.

[7] A. Brandt and D. Ron, *Chapter 1 : Multigrid solvers and multilevel optimization strategies*, in Multilevel Optimization and VLSICAD, J. Cong and J. R. Shinnerl, eds., Kluwer, 2003.

[8] W. L. Briggs, V. E. Henson, and S. F. McCormick, *A multigrid tutorial*, SIAM, 2000.

[9] A. Buluc and E. G. Boman, *Towards scalable parallel hypergraph partitioning*, in CSRI Summer Proceedings 2008, Sandia National Labs, 2008, pp. 109–119.

[10] A. BULUÇ, H. MEYERHENKE, I. SAFRO, P. SANDERS, AND C. SCHULZ, *Recent advances in graph partitioning*, in Algorithm Engineering, Springer, 2016, pp. 117–158.

[11] Ü. ÇATALYÜREK AND C. AYKANAT, *Patoh (partitioning tool for hypergraphs)*, in Encyclopedia of Parallel Computing, Springer, 2011, pp. 1479–1487.

[12] U. V. CATALYUREK AND C. AYKANAT, *Hypergraph-partitioning-based decomposition for parallel sparse-matrix vector multiplication*, IEEE Trans. Parallel Distrib. Syst., 10 (1999), pp. 673–693.

[13] U. V. CATALYUREK, E. G. BOMAN, K. D. DEVINE, D. BOZDAĞ, R. T. HEAPHY, AND L. A. RIESEN, *A repartitioning hypergraph model for dynamic load balancing*, J. Parallel Distrib. Comput., 69 (2009), pp. 711–724.

[14] T. CHAN, A. LOUIS, Z. G. TANG, AND C. ZHANG, *Spectral properties of hypergraph laplacian and approximation algorithms*, arXiv preprint arXiv:1605.01483, (2016).

[15] J. CHEN AND I. SAFRO, *Algebraic distance on graphs*, SIAM J. Sci. Comput., 33 (2011), pp. 3468–3490.

[16] T. A. DAVIS AND Y. HU, *The university of florida sparse matrix collection*, ACM Trans. Math. Software, 38 (2011), p. 1.

[17] D. DELLING, A. V. GOLDBERG, I. RAZENSHTEYN, AND R. F. WERNECK, *Graph partitioning with natural cuts*, in Parallel & Distributed Processing Symposium (IPDPS), 2011 IEEE International, IEEE, 2011, pp. 1135–1146.

[18] K. D. DEVINE, E. G. BOMAN, R. T. HEAPHY, R. H. BISSELING, AND U. V. CATALYUREK, *Parallel hypergraph partitioning for scientific computing*, in Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International, IEEE, 2006, pp. 10–pp.

[19] K. D. DEVINE, V. LEUNG, E. G. BOMAN, S. RAJAMANICKAM, L. A. RIESEN, AND U. CATALYUREK, *Zoltan user's guide, version 3.8.(2014)*, 2014.

[20] I. S. DHILLON, Y. GUAN, AND B. KULIS, *Weighted graph cuts without eigenvectors: A multilevel approach*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 29 (2007), pp. 1944–1957.

[21] C. M. FIDUCCIA AND R. M. MATTHEYSES, *A linear-time heuristic for improving network partitions*, in Papers on Twenty-five years of electronic design automation, ACM, 1988, pp. 241–247.

[22] M. FIEDLER, *Algebraic connectivity of graphs*, Czechoslovak mathematical journal, 23 (1973), pp. 298–305.

[23] M. R. GARY AND D. S. JOHNSON, *Computers and intractability: A guide to the theory of np-completeness*, 1979.

[24] L. HAGEN AND A. B. KAHNG, *New spectral methods for ratio cut partitioning and clustering*, IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., 11 (1992), pp. 1074–1085.

[25] W. W. Hager, J. T. Hungerford, and I. Safro, *A multilevel bilinear programming algorithm for the vertex separator problem*, Computational Optimization and Applications, (2017).

[26] B. Hendrickson and R. W. Leland, *A multi-level algorithm for partitioning graphs.*, SC, 95 (1995).

[27] 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, et al., *An overview of the trilinos project*, ACM Trans. Math. Software, 31 (2005), pp. 397–423.

[28] S. Hu and L. Qi, *Algebraic connectivity of an even uniform hypergraph*, J. Comb. Optim., 24 (2012), pp. 564–579.

[29] Y. F. Hu and J. A. Scott, *A multilevel algorithm for wavefront reduction*, SIAM J. Sci. Comput., 23 (2001), pp. 1352–1375.

[30] E. Ihler, D. Wagner, and F. Wagner, *Modeling hypergraphs by graphs with the same mincut properties*, Inf. Process. Lett., 45 (1993), pp. 171–175.

[31] E. John and I. Safro, *Single-and multi-level network sparsification by algebraic distance*, Journal of Complex Networks, 5 (2016), pp. 352–388.

[32] G. Karypis, R. Aggarwal, V. Kumar, S. Member, S. Shekhar, and S. Member, *Multilevel hypergraph partitioning: Application in vlsi domain*, in In Proceedings ACM/IEEE Design Automation Conference, 1997, pp. 526–529.

[33] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar, *Multilevel hypergraph partitioning: applications in vlsi domain*, IEEE Trans. Very Large Scale Integr. (VLSI) Syst., 7 (1999), pp. 69–79.

[34] G. Karypis and V. Kumar, *A fast and high quality multilevel scheme for partitioning irregular graphs*, SIAM J. Sci. Comput., 20 (1998), pp. 359–392.

[35] B. W. Kernighan and S. Lin, *An efficient heuristic procedure for partitioning graphs*, Bell Syst. Tech. J., 49 (1970), pp. 291–307.

[36] S. Lafon and A. B. Lee, *Diffusion maps and coarse-graining: A unified framework for dimensionality reduction, graph partitioning, and data set parameterization*, IEEE Trans. Pattern Anal. Mach. Intell., 28 (2006), pp. 1393–1403.

[37] J. Leskovec and A. Krevl, *SNAP Datasets: Stanford large network dataset collection.* http://snap.stanford.edu/data, June 2014.

[38] S. Leyffer and I. Safro, *Fast response to infection spread and cyber attacks on large-scale networks*, Journal of Complex Networks, 1 (2013), pp. 183–199.

[39] F. Lotfifar and M. Johnson, *A multi–level hypergraph partitioning algorithm using rough set clustering*, in European Conference on Parallel Processing, Springer, 2015, pp. 159–170.

[40] A. MISLOVE, M. MARCON, K. P. GUMMADI, P. DRUSCHEL, AND B. BHATTACHARJEE, *Measurement and Analysis of Online Social Networks*, in Proceedings of the 5th ACM/Usenix Internet Measurement Conference (IMC'07), San Diego, CA, October 2007.

[41] U. NAUMANN AND O. SCHENK, *Combinatorial Scientific Computing*, Chapman & Hall/CRC, 1st ed., 2012.

[42] A. Y. NG, M. I. JORDAN, Y. WEISS, ET AL., *On spectral clustering: Analysis and an algorithm*, in NIPS, vol. 14, 2001, pp. 849–856.

[43] D. A. PAPA AND I. L. MARKOV, *Hypergraph partitioning and clustering.*, 2007.

[44] D. RON, I. SAFRO, AND A. BRANDT, *Relaxation-based coarsening and multiscale graph organization*, Multiscale Modeling & Simulation, 9 (2011), pp. 407–423.

[45] I. SAFRO, D. RON, AND A. BRANDT, *Graph minimum linear arrangement by multilevel weighted edge contractions*, J. Algorithms, 60 (2006), pp. 24–41.

[46] I. SAFRO, P. SANDERS, AND C. SCHULZ, *Advanced coarsening schemes for graph partitioning*, ACM Journal of Experimental Algorithmics (JEA), 19 (2015), pp. 2–2.

[47] I. SAFRO AND B. TEMKIN, *Multiscale approach for the network compression-friendly ordering*, J. Discrete Algorithms, 9 (2011), pp. 190–202.

[48] P. SANDERS AND C. SCHULZ, *Engineering multilevel graph partitioning algorithms*, in ESA, C. Demetrescu and M. M. Halldórsson, eds., vol. 6942 of Lecture Notes in Computer Science, Springer, 2011, pp. 469–480.

[49] S. SCHLAG, V. HENNE, T. HEUER, H. MEYERHENKE, P. SANDERS, AND C. SCHULZ, *k-way hypergraph partitioning via n-level recursive bisection*, in 2016 Proceedings of the Eighteenth Workshop on Algorithm Engineering and Experiments (ALENEX), SIAM, 2016, pp. 53–67.

[50] C. STAUDT, A. SAZONOVS, AND H. MEYERHENKE, *Networkit: An interactive tool suite for high-performance network analysis*, CoRR, abs/1403.3005, (2014).

[51] A. STREHL AND J. GHOSH, *Cluster ensembles—a knowledge reuse framework for combining multiple partitions*, J. Mach. Learn. Res, 3 (2002), pp. 583–617.

[52] L. SUN, S. JI, AND J. YE, *Hypergraph spectral learning for multi-label classification*, in Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2008, pp. 668–676.

[53] A. TRIFUNOVIC, *Parallel algorithms for hypergraph partitioning*, University of London, 2006.

[54] B. VASTENHOUW AND R. H. BISSELING, *A two-dimensional data distribution method for parallel sparse matrix-vector multiplication*, SIAM review, 47 (2005), pp. 67–95.

[55] D. ZHOU, J. HUANG, AND B. SCHÖLKOPF, *Learning with hypergraphs: Clustering, classification, and embedding*, in NIPS, vol. 19, 2006, pp. 1633–1640.