Detection and Visualization of Dense Subgroups at Multiple Resolutions in Large Social Networks

Neil Gupta*, Joydeep Ghosh†, Gunjan Gupta*, Sheshank Shankar*, and Alex Tarasar*
*Lightsphere AI, Bellevue, WA, USA
neilgupta@lightsphereai.com, gunjankgupta@lightsphereai.com, sheshank@lightsphereai.com, alex@lightsphereai.com
†Department of Electrical and Computer Engineering, The University of Texas at Austin, Austin, TX, USA
jghosh@utexas.edu

***Abstract*—In large graphs such as those representing social or other affinity networks, one is often interested in finding dense clusters, i.e., subgraphs with relatively high connectivity, embedding in such graphs. Inspired by a classical approach called Hierarchical Mode Analysis, which works only for data in metric spaces, we introduce a novel algorithm called HIMAG (Hierarchical Incremental Mode Analysis for Graphs) that detects dense subgraphs at multiple resolutions, while ignoring "non-dense" areas. We also provide a powerful multi-resolution visualization tool customized for the new algorithm. We present results on two standard benchmark social graph datasets as well as a motivating real-world application, to show the power of our approach and compare it with some standard graph partitioning algorithms that were retrofitted to produce dense clusters by pruning non-dense data in a non-trivial manner. We are also open-sourcing the new dense graph datasets and tools to the community.**

***Index Terms*—graph learning, dense graphs, hierarchical clusters, visualization, automated cluster selection, social graphs, hierarchical mode analysis**

I. Introduction

In this paper, the goal is to find "dense" subgraphs (perhaps simultaneously at different resolutions) while pruning out a large fraction of irrelevant background data that may not cluster well. This setting was originally motivated by a real-world problem that we encountered. To concretize the issues being addressed, we describe this application before moving to the technical sections.

A. Real-time Social Viral Marketing

This motivating, real-world application involves finding clusters in a dataset of Instagram posts. The sparse graph consists of Instagram posts pre-filtered by an upstream crawler for a relevant product area (Wellness or Education). Each post is a node in the graph, and the weight of the edge between two posts depends on the degree of overlap in hashtags between the two posts, and is thresholded by some value. The dense clusters to be discovered either map to large groups of potential customers with a smaller number of more organic posts and shared interests, to whom we may want to send highly targeted messaging campaigns, or belong to single active influencers with a large number of highly related marketing posts, to whom we want to reach out to help with social viral marketing. Influencer and consumer groups naturally exist at varying degrees of "density" or cohesiveness depending on the type of consumers or influencers, but influencer clusters tend to be smaller and denser. Also, a large fraction of the posts may not cluster at all, and some of the clusters discovered may not be relevant for the product area and are easy to remove when separated out automatically.

IEEE/ACM ASONAM 2020, December 7-10, 2020
978-1-7281-1056-1/20/$31.00 © 2020 IEEE

B. Outline

To begin addressing the challenging setting described above, we start by first defining a flow-based [1] notion of density for nodes in a graph that we call *flow density* in Section III-C. The resolution or scale of flow density can be controlled by a similarity threshold. Then, we present a fast algorithm that can incrementally grow the most dense regions of the graph by varying this similarity threshold to discover subgraphs at variable densities/resolutions, presented in Section IV. In Section V-A, we present an edge-based measurement formulation that allows us to evaluate clusterings of dense subgraphs, including two new metrics: Edge ARI and Point Precision, both specially designed to work with edge-based partially labeled data. Our method enables pruning of "non-dense" portions of a graph while simultaneously selecting the most stable clusters at multiple resolutions, and this is the source of the substantially better results observed for HIMAG in our paper compared to other methods, on all five of the datasets studied in Section V-B. To help one see the results, we introduce a multi-resolution clustering visualization tool, Gene DIVER 3.0, which provides a user interface to browse the cluster hierarchy from HIMAG, in Section VI.

II. Related Work

A. Spatial Methods

For data given in a metric space, DBSCAN [2] is perhaps the most well known algorithm that can find dense clusters of arbitrary shapes. It does so by placing a fixed-radius spherical ball at each data point, and then counting the number of points falling within this spherical neighborhood. It then prunes all points that don't have enough neighbors (passed as a parameter) before performing clustering. However, DBSCAN cannot discover dense clusters at multiple resolutions/densities as it uses a single threshold for density; while nearby higher density clusters tend to merge into one cluster, lower density clusters tend to be fragmented or incomplete.

This problem of simultaneously finding dense clusters at multiple resolutions/densities was more successfully addressed by [3], and further unified with other related algorithms by [4]. [3], [4] are based on the concept of Hierarchical Mode Analysis (HMA), first proposed in [5], and the methods involve three major steps: (1) define density around each data point using a sphere, where the radius controls density threshold; (2) produce a hierarchy of clusters at multiple resolutions/densities by varying the radius thresholds; (3) select the most robust clusters identified over all resolutions by using the concept of stability proposed in [3]. However, as with DBSCAN, all these approaches require the availability of a meaningful embedding of the data in a metric space, and cannot readily apply to arbitrary similarity graphs.

B. Graph Partitioning Algorithms

While the algorithms mentioned above qualitatively provide the kinds of capabilities needed, alas none of these methods work on sparse graphs where only pairwise similarities between some data points is available. However, there do exist a host of clustering approaches that are based on graph partitioning, including hMETIS [6], KaHIP [7], [8], KaHyPar [9], [10], and PaToH [11]. hMETIS has stood the test of time and is still one of the most popular graph partitioning methods. KaHIP (Karlsruhe Fast Flow Partitioner) is a family of algorithms using flow-based partitioning and also supports parallelism [7]. KaHyPar is also a multilevel hypergraph partitioning algorithm utilizing an excess incremental breadth-first search (IBFS) maximum flow algorithm [12] and Boykov–Kolmogorov max-flow algorithm [13]. PaToH uses greedy hypergraph growing [14] and Boundary Fiduccia–Mattheyses algorithms [15]. Despite this rich literature and after extensive literature survey, we could not find any graph clustering or partitioning methods that could simultaneously discover clusters from graphs at multiple resolutions, while also naturally pruning out a large fraction of less cohesive parts of the graph. That was the motivation for the work presented herein. There are methods that can produce a hierarchical partitioning on graphs [16], [17], but cannot perform multi-resolution pruning of less cohesive regions. There is also a class of methods known as motif mining used to identify highly recurring small subgraphs [18], but they cannot find topologies or large structures. In order to make experimental comparisons, we took open-source versions of some leading graph partitioning algorithms and modified their results in a non-trivial way so that they could also prune out low-cohesiveness points. The results (Section V-B) show that even with these enhancements, the algorithms still do not perform as well as the approach introduced in this paper.

III. Preliminaries

A. Notation

Bold-faced lowercase variables, e.g. $\mathbf{x}$, represent vectors/arrays whose i^{th} element are accessed as $\mathbf{x}(i)$ (1-indexed). Arrays can contain sets as well as numbers. Calligraphic upper-case alphabets such as $\mathcal{X}$ represent sets, and can be notated with elements listed (e.g. $\{a, b, c\}$), with a predicate (e.g. $\{x \in \mathbb{R} | x < 7\}$), or enumerated with $\{x_i\}_{i=1}^{n}$ where x_i are the individual elements. $\mathbb{Z}^+$ represents the domain of positive integers. $\mathbb{R}$ and $\mathbb{R}^d$ represent the domain of real numbers and a d-dimensional vector space, respectively. Bold-faced capital letters such as $\mathbf{M}$ represent a two-dimensional matrix, which is accessed as $\mathbf{M}(i, j)$ (for row i, column j; both 1-indexed). A tuple is denoted by parentheses (e.g. (a, b, c)).

B. Spatial Density Estimation

Let $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^{n} \subsetneq \mathbb{R}^d$ be a set of d-dimensional data points that need to be clustered for a spatial dataset. Let $\mathbf{D}_S$ represent the corresponding $n \times n$ symmetric Euclidean distance matrix such that $\mathbf{D}_S(i, j)$ is the distance between $\mathbf{x}_i$ and $\mathbf{x}_j$. Given some $r_\epsilon \in \mathbb{R}^+$, the Auto-HDS algorithm [3] defines the spatial density $\rho_{r_\epsilon}(\mathbf{x}_i)$ that has the property $\rho_{r_\epsilon}(\mathbf{x}) \propto |\{\mathbf{y} \in \mathcal{X} | \mathbf{D}_S(\mathbf{x}, \mathbf{y}) \leq r_\epsilon\}|$.

This notion of density corresponds to a uniform sphere of radius r_ϵ as the kernel, since the points within the sphere contribute equally to the density irrespective of their distance from the point.

C. Flow Density of Nodes in Graph

Let $\mathbf{S}_\epsilon$, an $n \times n$ matrix, represent a sparse, weighted, undirected graph with n nodes. $\mathbf{S}_\epsilon(i, j) = 0$ if the similarity $\mathbf{S}(i, j)$ is less than a threshold, s_ϵ. Thus, we have $s_\epsilon < \mathbf{S}_\epsilon(i, j) \leq 1$ representing the edge weight between any two nodes i and j. Note that $\mathbf{S}_\epsilon$ is symmetric, and there are no self-loops. The *flow* between any two nodes a and b through a node i in the graph is computed as $\mathbf{S}_\epsilon(a, i)\mathbf{S}_\epsilon(i, b)$, and the total flow between nodes a and b, which is a standard computation for graphs [1], [19], is given by $flow_\epsilon(a, b) = \sum_{i=1}^{n} \mathbf{S}_\epsilon(a, i)\mathbf{S}_\epsilon(i, b)$.

We now introduce a novel notion of density for a node in a graph, that we call *flow density*, as simply the total flow to the node from every other node:

$$\rho_{flow_\epsilon}(i) = \sum_{j=1}^{n} flow_\epsilon(i, j) \tag{1}$$

This can be computed efficiently for a sparse graph, and can also be computed incrementally for varying s_ϵ—a property that is handy for HIMAG described in Section IV.

D. Properties & Motivation for Flow Density

If the non-zero edges in $\mathbf{S}_\epsilon$ were all set to 1 (an unweighted graph), then the flow between two nodes a and b represents the total number of shared neighbors between a and b. Though flow is the more commonly used term for describing this relationship, other terms such as link count [20], connectivity [19], and shared neighbor similarity [21] have also been used. These flow-based methods [19]–[22] provide explanations why flow is a better measure of true closeness between two nodes than similarity, and this concept is also incorporated in some of the graph partitioning methods we compare with (see Sections II and V-B). The reason for the improvement in performance

TABLE I
MARKETING EXPERIMENTS GRAPH SIZES AND PARAMETERS.

Segment	Education	Wellness
No. of nodes	54,836	35,308
No of edges	1,859,732	7,127,456
s_{EJ} **threshold Topic**	0.25	0.25
Node sample fraction	1.0	0.5

when using flow as opposed to direct neighbors is that noisy neighbors with strong similarity to a given node a that are not well-connected to other neighbors of a get a small flow with a. This results in easier discovery of true highly-connected subgraphs in the data. The flow however, is still defined only on pairs of nodes and not on nodes themselves, which is what we need to be able to prune non-dense nodes. This dilemma is resolved by the notion of flow density (Equation 1), which is a measure of the *flow neighborhood density* of the node, since it is performing a weighted count of the flow neighbors.

A crucial property of flow density, which we exploit in HIMAG, is the ability of the flow density to capture *local* flow density at a node when it is computed on a graph pre-thresholded with s_ϵ. When s_ϵ is small, the total flow through the many nodes that are not tightly connected to a dominates, creating a smoothing effect on the density. In contrast, when s_ϵ is large, the flow density of a is only contributed to by nodes that have a large similarity to a, which can be considered "nearby" nodes. Thus s_ϵ controls the resolution and smoothness of flow density variations in the graph visible to the clustering algorithm, and is exploited by HIMAG to find clusters at multiple resolutions. This is akin to the radius r_ϵ used to compute local density in the spatial HMA-style algorithms.

E. Benchmark Graphs & Datasets

Sim-2 Graph: Sim-2 is a synthetic 2D dataset sampled from 5 spherical Gaussians and an additional background distribution, described in [3]. converting this to a similarity graph using the distance between points gives us Sim-2 Graph.

Marketing Graphs: The real-world use-case that motivated our paper was the problem of automatically discovering influencers and potential consumer networks for any given application domain for a digital marketing application. This data is built using an upstream intelligent crawler that crawls all public posts on Instagram (www.instagram.com) containing a set of hashtags for a given segment, which is optimized automatically by upstream AI that is beyond the scope of this paper, but it is important to know that this results in highly domain-specific Instagram posts for each segment. In this paper we focus on two domains: Wellness and Education. For our use-case, the final experiments were performed on graphs with sizes shown in Table I.

We then use the Min-Max Similarity (MMSIM) [19], an extension of the Jaccard coefficient, on posts to compute the similarity graph. We also thresholded the graphs to sparsify them a bit for computational speed and quality.

Standard social network benchmarks: We present results on two standard anonymized social network datasets: Pokec [23], Slovakia's largest social network, and the LiveJournal social network [24]. These datasets provide us with two independent signals that help us benchmark without the need for human labeling. The first signal is the social graph itself, which is an unweighted graph of connections between users. The second signal is an independent undirected weighted *interests graph* built by connecting people who share "interests". We use the community profile that the datasets provide, namely "hobbies" and "communities" for Pokec and LiveJournal respectively, to compute this graph. We build the interests graph edges using Min-Max Similarity [19] on shared interests between two users.

To set up the prediction problem, we first intersected the social graph nodes with the interests graph nodes, and then split the social graph into a training and measurement set; the training set is 2% subset of random edges sampled from the full social graph, while we hold the remaining 98% of the remaining social graph blind to our clustering algorithms to see how well they can predict these missing connections. We also make the training graph weighted (as the social graph is unweighted) by simply averaging the edge weights with the interests graph. This provides us with a partially weighted training graph where the edges range between 0.5 and to 1.0 depending upon how many interests the connections have in common. Table II details some of the different intermediate graphs extracted from the external datasets, and Table III summarizes the final graphs generated from our datasets.

Note that we expected not all friends to necessarily have many hobbies in common, and conversely, many of the connections being predicted are likely good matches, but may not be already connected in the measurement set. Of course, for our setup, we can only measure against people who are *already* connected in the measurement set; it would be interesting to use this method as a powerful setup for recommending new connections for popular social networks such as LinkedIn (www.linkedin.com) or Facebook (www.facebook.com), and see how it performs.

IV. HIMAG

Our new *Hierarchical Incremental Mode Analysis for Graphs* algorithm outputs an HMA matrix like the one described in [3], then we follow a similar process to [3] to identify clusters across resolutions and relabel them, dictionary sort the rows (nodes) for better visualization, and assign clusters a stability using Equation 2.

For experimental evaluation of our algorithm, we removed specific clusters based on the stability values to get a non-overlapping clustering, but if one desires a hierarchical clustering this step can be omitted.

A. Graph HMA

HIMAG works by redefining the key notions of spatial radius (r_ϵ), spatial density (ρ_ϵ), and density threshold (n_ϵ)

TABLE II
SOCIAL EXPERIMENTS GRAPH SIZES AND PARAMETERS.

Social Network	**Pokec**		**LiveJournal**	
No. of interests	76,914 hobbies		287,512 communities	
Top Interests	**Count**	**Freq. Thresh.**	**Count**	**Freq. Thresh.**
Top Interests	12,285	2	1,489	500
Graph sizes	**Node count**	**Edge count**	**Node count**	**Edge count**
Social Graph	1,632,803	30,622,564	3,997,962	34,681,189
Top Interests Graph	758,054	10 billion+	612,577	20 billion+
Intersected Graph	736,120	22,411,849	612,577	10,070,149
Training Graph	208,620	151,336	212,076	201,158
Measurement Graph	711,950	7,420,344	612,007	9,868,991
Training Social Graph Fraction	2%		2%	

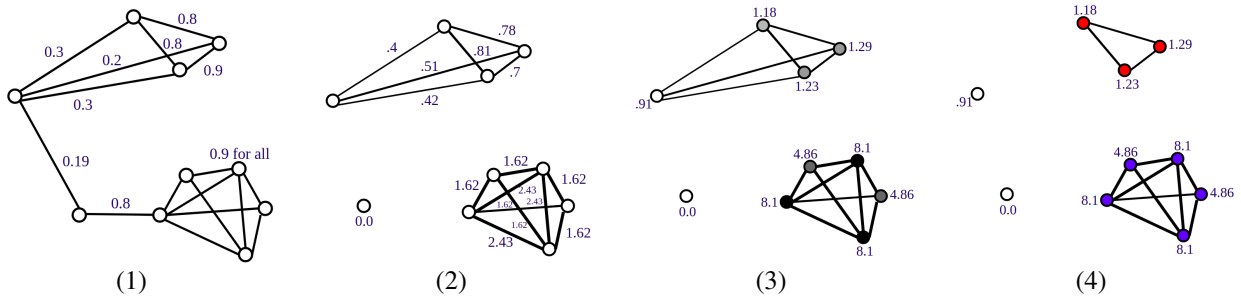


Fig. 1. A simplified illustration of one level of HIMAG Graph clustering showing (1) a sparse undirected weighted graph, (2) kept all edges for $s_\epsilon > 0.2$, compute edge flow, (3) compute node density ρ_{flow} and prune non-dense points with $\rho_{flow} < n_{flow_\epsilon}$, (4) cluster all connected dense components and prune non-dense points.

TABLE III
BENCHMARK GRAPH DATASETS, THEIR APPLICATION DOMAIN, AND THE SIZES OF GRAPHS CONSTRUCTED FROM THEM.

Dataset	**Application**	**No. of nodes**	**No. of edges**
Sim-2 Graph	Simulated	1,304	848,253
Wellness	Marketing	35,308	7,127,456
Education	Marketing	54,836	1,859,732
Pokec	Social	758,054	10 billion+
LiveJournal	Social	612,577	10 billion+

used in the Spatial HMA algorithms [3], [4], [25]. As described in Section III-D, and illustrated in Figure 1, given a weighted, normalized, undirected similarity graph $\mathbf{S}$ with n nodes, we first compute the total flow ρ_{flow_ϵ} to each node in a thresholded similarity graph $\mathbf{S}_\epsilon$ using the similarity threshold s_ϵ. Then, we take all nodes i with $\rho_{flow_\epsilon}(i) > n_{flow_\epsilon}$, and cluster them using the well-known flood fill method, where nodes i and j cluster together if $\mathbf{S}_\epsilon(i,j) > 0$. Thus, each connected component of $\mathbf{S}_\epsilon$ (with non-dense nodes removed) becomes a cluster.

We compute clusterings at multiple resolutions by varying s_ϵ. This gives us an $n \times l$ HMA matrix $\mathbf{L_{HMA}}$, where each row represents a node, and each column corresponds to one clustering level, associated with a specific threshold s_ϵ. The values in the matrix are cluster IDs, which are generated independently for each level.

The list of values to use for s_ϵ is generated based on the input similarity matrix $\mathbf{S}$. [3] calculates values of r_ϵ so that the ratio of points clustered between two successive levels remains a constant r_{shave}. For a graph this would be difficult to compute, so we instead calculate the values of s_ϵ so that the ratio of the number of nonzero edges in the thresholded similarity matrix $\mathbf{S}_\epsilon$ between two successive levels remains a constant r_{shave}.

B. HIMAG Algorithm

The HIMAG algorithm provides a computationally efficient way to compute the Graph HMA matrix.

We store the graphs not as matrices, but as sets of edge tuples (i, j, s), each of which represents an edge between nodes i and j with weight s. This sparse representation saves memory and allows us to perform some optimizations.

First, to compute the list of values of s_ϵ to be used, we sort all the edges in the input graph $\mathbf{S}$ into a list, and take evenly spaced samples (and remove duplicates). This list can also easily be partitioned to produce buckets of edges between successive thresholds. Thus, as we lower the similarity threshold s_ϵ, we have stored in memory the set of *new* edges introduced to $\mathbf{S}_\epsilon$ at each level.

We then compute the total flow to each node ($\rho_{flow_\epsilon}(i)$) incrementally as we lower s_ϵ and add new edges to $\mathbf{S}_\epsilon$. We also keep track of the neighbors of each node i in a data structure $\mathbf{e_{nbr}}$ where $\mathbf{e_{nbr}}(i) = \{j \in \mathbb{Z}^+ | j < n \wedge \mathbf{S}_\epsilon(i,j) > 0\}$, and we keep track of the nodes that are clustered (i.e. their flow density is above the threshold n_{flow_ϵ}) in a set of clusters $\mathcal{C}$.

Let's look at the addition of a single new edge (a, b, s), and how it affects the values of ρ_{flow_ϵ} and $\mathbf{e_{nbr}}$. This edge connects nodes a and b with a weight of s. All of a's current neighbors will gain flow with node b through a, and vice versa. So $\forall i \in \mathbf{e_{nbr}}(a)$, we increment $\rho_{flow_\epsilon}(i)$ and $\rho_{flow_\epsilon}(b)$ by

$\mathbf{S}_\epsilon(i,a)\mathbf{S}_\epsilon(a,b)$, and vice versa for b's neighbors. Finally, we add node a to $\mathbf{e_{nbr}}(b)$ and node b to $\mathbf{e_{nbr}}(a)$.

As we add edges, we keep track of which nodes' flow density increased that were not already clustered. After adding all the edges in one bucket, we check if any of these nodes i now have a flow density $\rho_{flow_\epsilon}(i)$ above the flow density threshold n_{flow_ϵ}. If they do, we add them to a cluster in $\mathcal{C}$ according to the connected components rules, merging clusters when they get connected, or creating new ones as necessary.

After this, the clustering $\mathcal{C}$ at the current level k is saved into the HMA matrix $\mathbf{L_{HMA}}$. Each of the clusters in $\mathcal{C}$ is assigned a positive integer ID, and then each node i in the cluster has $\mathbf{L_{HMA}}(i,k)$ set to this ID. Nodes i that are not clustered at level k therefore have $\mathbf{L_{HMA}}(i,k) = 0$.

Since this creates the HMA hierarchy by growing and merging clusters, as opposed to shaving and splitting them, the process can be stopped early and still yield meaningful results, further saving computation time. This is useful for many practical applications where the denser subset of the topology discovered is more important.

C. HMA Visualization & Cluster Selection

Since the HMA matrix from HIMAG has an identical form to spatial Auto-HDS, we follow mostly the same process as [3] for visualization, but cluster stability is defined slightly differently; it measures the stability of a cluster as proportional to the fraction of data (nodes) shaving a cluster survives. If $f_C(start_i)$ and $f_C(end_i)$ represent the fraction of nodes clustered in total when the cluster $\mathcal{C}_i$ first comes into existence and when it disappears, in the order from left to right in the $\mathbf{L_{HMA}}$ matrix (shrinking clusters), then we define *stability* as:

$$Stab(\mathcal{C}_i) = \frac{\log(f_C(start_i)) - \log(f_C(end_i))}{\log(0.99)} \qquad (2)$$

This notion of stability improves [3] in that the denominator is set to $\log(0.99)$, corresponding to an r_{shave} of 1%—an arbitrary normalization, instead of the variable $\log(1-r_{shave})$. As noted in [26], this makes the notion of stability independent of the node shaving rate, which is necessary for HIMAG given its edge-based r_{shave}. Note that it is trivial to modify HIMAG to compute all the levels, as the incremental nature of the algorithm minimizes computational overhead for this. However, the larger HMA matrix would hurt the algorithm's space complexity, for only a small gain in cluster resolution.

D. Time & Space Complexity

The time complexity of HIMAG can be shown to be $O(E\log(E) + Ek_n)$, where E is the number of edges in the graph, and k_n is a node's average number of neighbors.

The space complexity of HIMAG when storing the HMA matrix in memory is $O(E+n\log(E))$, where n is the number of nodes in the graph. If the HMA matrix is streamed directly to secondary storage, storing only two columns at a time in memory, the space complexity drops to $O(E+n)$.

TABLE IV
POINT PRECISION PERFORMANCE COMPARISON FOR THE MARKETING SEGMENTS COMPARING HIMAG WITH FLOW (F) VARIATION OF HMETIS FOR THE TOP 50 (BY STABILITY) AND ALL JUDGED CLUSTERS. HIMAG RECALL FOR ALL CLUSTERS FOR WELLNESS WAS 0.201 WITH $k = 162$ CLUSTERS, WHILE FOR EDUCATION THE RECALL WAS 0.218 WITH $k = 424$ CLUSTERS. FOR HMETIS, THE NUMBER OF CLUSTERS WAS DOUBLE WITH THE SAME RECALL. ALSO SHOWN IS THE NUMBER OF POINTS IN THE SMALLEST CLUSTER (S) AND THE LARGEST CLUSTER (B).

Algorithm	Wellness				Education			
	Top 50	All	S	B	Top 50	All	S	B
HIMAG	.9394	.9438	7	1388	.9973	.9748	2	248
hMETIS 2k, F	.7665	.718	7	159	.8953	.7336	2	87

V. EXPERIMENTAL EVALUATION

A. Evaluation Metrics

We present results using two metrics: Edge ARI and Point Precision. The Edge ARI used in this paper is a modification of the standard Adjusted Rand Index [27] that was designed for partitional clustering with edge-based labels.

Given a clustering $\mathcal{C}$ containing clusters $\mathcal{C}_i$ which each contain all pairs (a,b) representing an edge between two nodes a and b where a and b are clustered together in cluster i, and an exhaustive edge-based label set $\mathcal{L}_\mathcal{S}$ containing all pairs (a,b) that are linked ("must-link"), we define $\mathcal{C}_p = \{(a,b)|\exists\mathcal{C}_i \in \mathcal{C}; (a,b) \in \mathcal{C}_i\}$ as the set of all edges predicted as linked in a clustering $\mathcal{C}$.

If all the must-link edges found within a cluster $\mathcal{C}_i$ are assumed to be coming from a single label cluster of size c, then the number of must-link edges would be given by $\binom{c}{2} = \frac{c(c-1)}{2}$. Conversely, given the set of must-link edges within a cluster $(\mathcal{C}_i \cap \mathcal{L}_\mathcal{S})$, this hypothetical c can be calculated with $c = \frac{1}{2}\left(1 + \sqrt{8\left|\mathcal{C}_i \cap \mathcal{L}_\mathcal{S}\right| + 1}\right)$, the inverse of the quadratic equation for c. Point Precision can then be calculated as the weighted sum of the precision of the clusters, as $P_p\left(\mathcal{C}, \mathcal{L}_\mathcal{S}\right) = \sum_{\mathcal{C}_i \in \mathcal{C}} \frac{\frac{1}{2}\left(1 + \sqrt{8\left|\mathcal{C}_i \cap \mathcal{L}_\mathcal{S}\right| + 1}\right)}{\left|\mathcal{C}_i\right|}$.

For computing *Edge ARI*, an exhaustive label set is also required. Any edge not in $\mathcal{L}_\mathcal{S}$ is assumed to be "cannot-link" (should not link). The formula takes the form $\frac{Index - E(Index)}{Max(Index) - E(Index)}$, like the Adjusted Rand Index [27]. The index is simply the number of edges that are "must-link" and are predicted in the clustering $\mathcal{C}$. The expected index is the number of edges predicted multiplied by the proportion of total edges that are "must-link". The max index is whichever is smallest between the number of edges that are "must-link", and the number of edges predicted in the clustering $\mathcal{C}$, as both of these would have to be true for the edge to count towards the index. This gives us the formula, where n is the total number of nodes in the labeled graph:

$$ARI_E\left(\mathcal{C}, \mathcal{L}_\mathcal{S}\right) = \frac{\left|\mathcal{C}_p \cap \mathcal{L}_\mathcal{S}\right| - \left|\mathcal{C}_p\right| \frac{\left|\mathcal{L}_\mathcal{S}\right|}{\binom{n}{2}}}{\min\left(\left|\mathcal{C}_p\right|, \left|\mathcal{L}_\mathcal{S}\right|\right) - \left|\mathcal{C}_p\right| \frac{\left|\mathcal{L}_\mathcal{S}\right|}{\binom{n}{2}}}.$$

B. Results

Note that all methods received exactly the same input graph for each dataset. More details, along with the source code for all algorithms and measurements are provided with a supplement link to ensure these experiments are fully reproducible.

The number of clusters to predict (**k**) and the fraction of data clustered are passed to other methods based on the output of HIMAG to make the comparison fair. We also ran other methods with double the clusters (**2k**) to improve their performance in cases where they performed poorly compared to HIMAG otherwise. Since HIMAG clusters only a fraction of data, two different methods were used to make the partitional algorithms behave similarly. **RAND** represents randomly selecting a fraction of the points from each cluster, while **FLOW** represents shaving each cluster by removing the least flow-dense nodes until the desired number of nodes remain.

Table IV shows results on our new Marketing dataset, while Table V shows results on the other datasets. Note how HIMAG finds clusters of a more diverse size, and consistently outperforms other algorithms in the metrics, at various levels of clustering, which is also clearly shown in Figure 2.

VI. Gene DIVER Visualization & Tools

We developed Gene DIVER 3.0 for our experimental evaluation, as an extension to Gene DIVER 2.0 released in [3]. It now visualizes the cluster hierarchy produced by HIMAG, in addition to the spatial Auto-HDS already supported by Gene DIVER 2.0. It also comes with some other new capabilities. The source code for Gene DIVER 3.0 along with all our tools for the experimental setup, more detailed discussion of the results, and the four new datasets are available on GitHub[1].

The HMA hierarchy created by HIMAG, visualized by Gene DIVER, shows the clusters at various densities. The y-axis represents the different points, while the x-axis represents the shaving level or resolution/density. At each resolution, the colored areas represent the clusters while the black areas represent the pruned background points that do not cluster.

A. Sim-2

As described in Section III-E, the Sim-2 dataset is a synthetic 2D dataset sampled from 5 spherical Gaussians and an additional background distribution (Figure 3 (a) & (d)). The Sim-2 Graph dataset (Figure 3 (b)) is generated from Sim-2 with a simple transformation. This dataset not only allows us to more easily test and validate the HIMAG algorithm, but it is also useful for illustrative purposes.

On the HMA hierarchy produced by HIMAG for the Sim-2 Graph dataset (Figure 3 (c)), the small teal cluster at the very bottom corresponds to the magenta cluster in the spatial labels (Figure 3 (d)). Meanwhile, the long and thin light green cluster in the HMA hierarchy corresponds to the very dense dark blue cluster in the labels. This illustrates HIMAG's ability to detect clusters at varying resolutions. Gene DIVER can also show the high cluster precision of each cluster, which can be viewed by clicking on the clusters in the hierarchy.

[1] github.com/nog642/himag-release-asonam

B. Marketing

A good example of the topological relationships between the clusters is shown in Figure 4 (c). The parent cluster (outlined in white) contains 178 posts about the Law School Admissions Test (LSAT). The bottom gold child cluster contains 12 posts about practice LSAT questions. The light purple cluster directly above the gold cluster 27 posts with the hashtag "#lsatprep". The dark purple cluster directly above the light purple cluster contains 33 posts with the hashtag "#lsatstudying". The light brown child cluster on the top could not be analyzed because the posts have since been removed. These topological relationships cannot be found with traditional graph partitioning algorithms.

Running the HIMAG algorithm on the Wellness dataset produced pure clusters of various sizes. Images e–h of Figure 4 illustrate the different classifications of marketing clusters in the Wellness segment. Some clusters can be classified as commercial influencers—for example, one cluster of this kind contains posts from a company marketing their weight-loss tea supplement (example shown in Figure 4 (f)), with 15 points, and a cluster precision of 100%. Other clusters can be classified as individual influencers, such as one cluster of infographic posts from a diet and workout coach (example shown in Figure 4 (g)), with 60 points, and a purity of 100%. Lastly, some clusters can be classified as large communities with posts from many different users, like a cluster of posts of weight-loss tips in Portuguese (example shown in Figure 4 (h)), with 46 points, and a cluster precision of 100%.

The Instagram URL of each post is embedded in the dataset. Clicking the data point in Gene DIVER opens the Instagram post in a web browser, allowing for fast analysis.

VII. Conclusion

When we started this research, we had to find a way to discover dense clusters at multiple resolutions in an unsupervised manner, especially for the marketing real-world application, as new clusters arise in such data every week. The clusters had to be reasonably complete (high recall) and accurate (high precision). As can be seen clearly, our method's performance far exceeds those of other methods, even after adapting them to prune out non-dense regions in a principled way. With HIMAG, we find a large and diverse set of clusters of varying sizes and cohesiveness. We believe that what we have discovered with flow density is a deceptively simple concept, but one which is a fundamental breakthrough in several ways—how we should think about dense regions in graphs to elucidate dense clusters at multiple resolutions, why such a notion is not just the dominion of spatial density based methods derived from HMA, and why we don't have to think about flow on graphs only in terms of edges and pairwise nodes as graph partitioning methods have done until now.

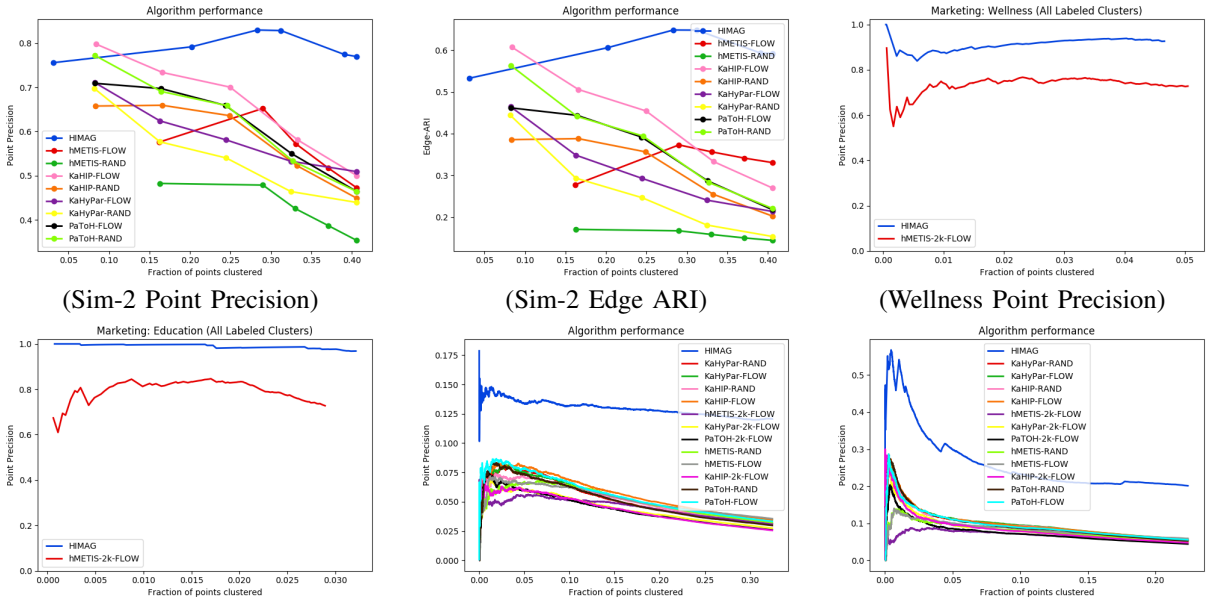
Acknowledgment

This research was supported by Lightsphere AI. We thank Andrew Galloway and Mary Remya for helping us build the machinery for marketing data, and Alex Mallen for helping

TABLE V

RESULTS ON POKEC, LIVEJOURNAL, & SIM-2 GRAPH BY POINT PRECISION (PP) AND EDGE ARI (E-ARI). THE RUNTIMES (RT) OF THE ALGORITHMS ARE SHOWN IN SECONDS ON AN 8-CORE LENOVO THINKSERVER MACHINE WITH 32 GB RAM. ALSO SHOWN IS THE NUMBER OF POINTS IN THE SMALLEST CLUSTER (S) AND THE LARGEST CLUSTER (B), AND THE NUMBER OF CLUSTERS (C). F IS SHORT FOR FLOW AND R IS SHORT FOR RAND IN THE ALGORITHM NAMES.

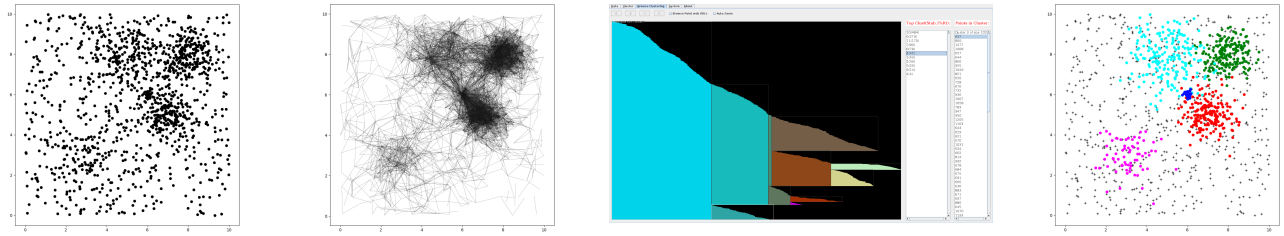
Algorithm	Pokec						LiveJournal						Sim-2					
	PP	E-ARI	RT	S	B	C	PP	E-ARI	RT	S	B	C	PP	E-ARI	RT	S	B	C
HIMAG	.121	.0214	147.88	3	74	8769	.201	.2541	343.18	3	488	6002	.770	.5919	68.4	20	223	6
hMETIS 2k, F	.031	.0102	58.34	2	13	17538	.055	.0229	77.168	2	71	11994	.479	.3107	52.9	36	76	10
KaHIP 2k, F	.026	.0089	28.09	2	5	17538	.048	.0187	92.58	2	6	12004	.482	.2395	45.71	43	44	10
hMETIS k, F	.035	.0057	58.34	2	25	8769	.059	.0167	77.16	2	108	6002	.493	.3307	52.9	46	211	5
KaHIP k, F	.034	.0055	28.09	3	9	8769	.057	.0141	92.58	3	10	6002	.500	.2695	45.71	95	109	5
hMETIS k, R	.032	.0052	58.34	2	24	8769	.055	.0157	77.16	2	108	6002	.380	.1660	52.9	45	212	5
KaHIP k, R	.031	.0050	28.09	4	9	8769	.056	.0142	92.58	3	10	6002	.449	.2024	45.71	96	109	5
KaHyPar 2k, F	.027	.0097	161.58	2	4	17538	.049	.0198	177.38	2	5	12004	.497	.2477	199.02	51	54	10
PaTOH 2k, F	.026	.0091	15.40	2	4	17538	.044	.0178	13.02	2	5	12004	.508	.2662	2.54	51	54	10
KaHyPar k, F	.035	.0057	161.58	4	8	8769	.057	.0142	177.38	4	9	6002	.509	.2136	199.02	104	107	5
PaTOH k, F	.033	.0054	15.40	4	8	8769	.056	.0137	13.02	4	9	6002	.464	.2176	2.54	104	107	5
KaHyPar k, R	.031	.0051	161.58	4	8	8769	.053	.0133	177.38	4	9	6002	.439	.1537	199.02	104	107	5
PaTOH k, R	.030	.0048	15.40	4	8	8769	.052	.0134	13.02	4	9	6002	.464	.2207	2.54	103	107	5



(Sim-2 Point Precision) (Sim-2 Edge ARI) (Wellness Point Precision)

(Education Point Precision) (Pokec Point Precision) (LiveJournal Point Precision)

Fig. 2. Edge ARI and Point Precision plots for Sim-2, Marketing, Pokec, & LiveJournal datasets. x-axis is recall when sorting and selecting most stable clusters using flow method for all algorithms, except for HIMAG which has its own stability.



(a) (b) (c) (d)

Fig. 3. Sim-2 synthetic dataset. (a) Unlabeled points in 2D euclidean space as seen by spatial clustering algorithm; (b) a sample of edges with $s_\epsilon \geq 0.9$ from the Sim-2 Graph data; (c) full Gene DIVER view of HMA hierarchy produced by HIMAG shows visually identical topology and clusters to spatial clustering; (d) labeled 2D data points used for measurements showing 5 labeled *dense* clusters in color, and the background (noise) points in black.

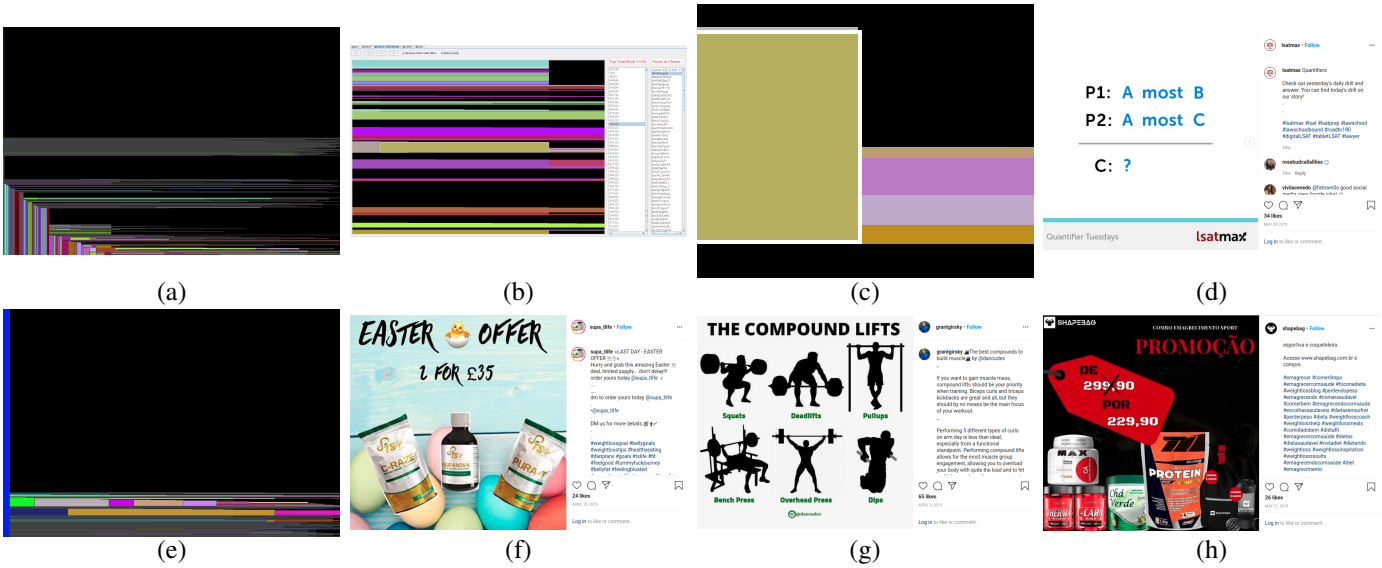


Fig. 4. Marketing Education dataset: (a) full HMA hierarchy produced by HIMAG for the Marketing Education dataset; (b) zoomed in full Gene DIVER view of HMA hierarchy produced by HIMAG; (c) example of the LSAT topology in HMA hierarchy; (d) post that is part of the "LSAT practice question" child cluster of the LSAT cluster. Marketing Wellness dataset: (e) full HMA hierarchy produced by HIMAG for the Marketing Wellness dataset; (f) advertisement for an Easter sale for a weight-loss supplement; (g) infographic by a diet and workout coach explaining the best types of compound lifts to build muscle; (h) promotion of a weight-loss powder in Portuguese.

with the early research and prototyping. We are also grateful to Ashley Prietto, Sandrine Gupta, and Kevin McCrea for helping us generate labels for measurements.

REFERENCES

[1] G. T. Heineman, G. Pollice, and S. Selkow, *Algorithms in a Nutshell*. O'Reilly Media, 2008, ch. 8, pp. 226–250.

[2] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. 2nd Int. Conf. Knowl. Discovery and Data Mining*, ser. KDD'96, 1996.

[3] G. Gupta, A. Liu, and J. Ghosh, "Automated hierarchical density shaving: A robust automated clustering and visualization framework for large biological data sets," *IEEE/ACM Trans. Computational Biol. and Bioinf.*, vol. 7, no. 2, pp. 223–237, 2010.

[4] R. J. G. B. Campello, D. Moulavi, A. Zimek, and J. Sander, "Hierarchical density estimates for data clustering, visualization, and outlier detection," *ACM Trans. Knowl. Discov. Data*, vol. 10, no. 1, Jul. 2015.

[5] D. Wishart, "Mode analysis: A generalization of nearest neighbour which reduces chaining effects," in *Proc. Colloquium in Numerical Taxonomy*. University of St. Andrews, Fife, Scotland: Academic Press, September 1968, pp. 282–308.

[6] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar, "Multilevel hypergraph partitioning: Application in VLSI domain," in *Proc. 34th Annu. Des. Automat. Conf.*, ser. DAC '97, 1997, pp. 526–529.

[7] P. Sanders and C. Schulz, "Kahip v0.6 – karlsruhe high quality partitioning."

[8] ——, "Think locally, act globally: Highly balanced graph partitioning," in *Experimental Algorithms, 12th Int. Symp., SEA 2013, Proc.*, vol. 7933. Springer, 2013, pp. 164–175.

[9] S. Schlag, C. Schulz, D. Seemaier, and D. Strash, "Scalable edge partitioning," in *2019 Proc. Meeting on Algorithm Engineering and Experiments (ALENEX)*, pp. 211–225.

[10] T. Heuer, P. Sanders, and S. Schlag, "Network Flow-Based Refinement for Multilevel Hypergraph Partitioning," in *17th Int. Symp. on Experimental Algorithms (SEA 2018)*, 2018, pp. 1:1–1:19.

[11] Ü. V. Çatalyürek and C. Aykanat, "PaToH: Partitioning Tool for Hypergraphs," pp. 1–33, November 1999.

[12] A. V. Goldberg, S. Hed, H. Kaplan, P. Kohli, R. E. Tarjan, and R. F. Werneck, "Faster and more dynamic maximum flow by incremental breadth-first search," in *Algorithms - ESA 2015*, N. Bansal and I. Finocchi, Eds. Springer, 2015, pp. 619–630.

[13] Y. Boykov and V. Kolmogorov, "An experimental comparison of min-cut/max- flow algorithms for energy minimization in vision," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 26, no. 9, pp. 1124–1137, 2004.

[14] U. V. Catalyurek and C. Aykanat, "Hypergraph-partitioning-based decomposition for parallel sparse-matrix vector multiplication," *IEEE Trans. Parallel and Distrib. Syst.*, vol. 10, no. 7, pp. 673–693, 1999.

[15] V. Olshevsky and E. Tyrtyshnikov, *Matrix Methods: Theory, Algorithms and Applications*. WORLD SCIENTIFIC, 2010.

[16] N. Tremblay and P. Borgnat, "Graph wavelets for multiscale community mining," *IEEE Trans. Signal Process.*, vol. 62, no. 20, 2014.

[17] J.-C. Delvenne, S. N. Yaliraki, and M. Barahona, "Stability of graph communities across time scales," *Proc. National Academy of Sciences*, vol. 107, no. 29, pp. 12 755–12 760, 2010.

[18] N. Kashtan, S. Itzkovitz, R. Milo, and U. Alon, "Efficient sampling algorithm for estimating subgraph concentrations and detecting network motifs," *Bioinformatics*, vol. 20, no. 11, pp. 1746–1758, 03 2004.

[19] G. K. Gupta and J. Ghosh, "Value-balanced agglomerative connectivity clustering," in *Data Mining and Knowledge Discovery: Theory, Tools, and Technology III*, vol. 4384. SPIE, 2001, pp. 6–15.

[20] S. Guha, R. Rastogi, and K. Shim, "ROCK: a robust clustering algorithm for categorical attributes," in *Proc. 15th Int. Conf. Data Engineering*, 1999, pp. 512–521.

[21] L. Ertöz, M. Steinbach, and V. Kumar, "Finding clusters of different sizes, shapes, and densities in noisy, high dimensional data," in *Proc. 2003 SIAM Int. Conf. Data Mining*, pp. 47–58.

[22] L. Bohlin, D. Edler, A. Lancichinetti, and M. Rosvall, "Community detection and visualization of networks with the map equation framework," in *Measuring Scholarly Impact*. Springer, 2014, pp. 3–34.

[23] L. Takac and M. Zábovský, "Data analysis in public social networks," *Int. Scientific Conf. and Int. Workshop Present Day Trends of Innovations*, pp. 1–6, 2012.

[24] J. Leskovec, K. J. Lang, A. Dasgupta, and M. W. Mahoney, "Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters," *Internet Mathematics*, vol. 6, no. 1, 2009.

[25] L. McInnes and J. Healy, "Accelerated hierarchical density based clustering," in *2017 IEEE Int. Conf. on Data Mining Workshops (ICDMW)*, pp. 33–42.

[26] R. J. G. B. Campello, D. Moulavi, and J. Sander, "A simpler and more accurate AUTO-HDS framework for clustering and visualization of biological data," *IEEE/ACM Trans. Computational Biol. and Bioinf.*, vol. 9, no. 6, pp. 1850–1852, 2012.

[27] L. Hubert and P. Arabie, "Comparing partitions," *Journal of Classification*, vol. 2, no. 1, pp. 193–218, 1985.