Robust Unsupervised Mining of Dense Sub-Graphs at Multiple Resolutions

Neil Gupta*
neilgupta@lightsphereai.com
Lightsphere AI
Bellevue, Washington

Gunjan Gupta*
gunjankgupta@lightsphereai.com
Lightsphere AI
Bellevue, Washington

Joydeep Ghosh
jghosh@utexas.edu
Dept. of ECE, UT Austin
Austin, Texas

Sheshank Shankar*
sheshank@lightsphereai.com
Lightsphere AI
Bellevue, Washington

Alex Mallen
ATMallen8@gmail.com
Bellevue, Washington

ABSTRACT

Category: Novel research paper. Whereas in traditional *partitional* clustering, each data point belongs to a cluster, there are several applications where only some of the points form relatively homogenous or "dense" groups, and points that don't seem to belong to any cluster need to be ignored. Moreover, different clusters may emerge at different scales or density levels. This makes it difficult to identify them using a single density threshold, especially if we also want to ignore the non-clustering data. If data is represented in a metric space, then recent extensions of a classical approach called Hierarchical Mode Analysis (HMA) are able to identify clusters at multiple resolutions, while ignoring "non-dense" areas. However, this approach does not apply when the relations between pairs of data points can only be represented as a (sparse) similarity or affinity graph. Motivated by two complex, real-life applications where one needs to identify dense subgraphs at multiple resolutions, while ignoring nodes that are not well connected in the similarity graph, we introduce a novel algorithm called HIMAG (Hierarchical Incremental Mode Analysis for Graphs) that provides capabilities analogous to HMA based methods but applicable to graphs. We also provide a powerful multi-resolution visualization tool customized for the new algorithm. We present results on the two motivating real-world applications as well as two standard benchmark social graph datasets, 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.

CCS CONCEPTS

• **Computing methodologies** → **Unsupervised learning**; **Cluster analysis**; *Topic modeling*; • **Mathematics of computing** → **Graph algorithms**; **Network flows**; **Paths and connectivity problems**; **Graph enumeration**; • **Information systems** → **Social networks**.

KEYWORDS

graph learning, dense graphs, hierarchical clusters, visualization, automated cluster selection, social graphs, hierarchical mode analysis

ACM Reference Format:
Neil Gupta, Gunjan Gupta, Joydeep Ghosh, Sheshank Shankar, and Alex Mallen. 2020. Robust Unsupervised Mining of Dense Sub-Graphs at Multiple Resolutions. In *Proceedings of MLG2020 (16th International Workshop on Mining and Learning with Graphs) (MLG2020).* ACM, New York, NY, USA, 8 pages. https://doi.org/TBD

1 INTRODUCTION

In classical clustering, each data point is assigned to at least one cluster based on how close or similar that point is to that cluster, compared to other clusters. However, in many real-world problems, only a small subset of the data points cluster into relatively homogenous groups, while the rest show little or no clustering tendencies. This could be due to the fact that only a subset of the points actually cluster well into highly cohesive or "dense" groupings, while the rest can be treated as a "don't care" set. Additionally, these clusters could occur at different "resolutions" or densities. For such problems, could we use traditional clustering algorithms to simply cluster some fraction of the data, and then prune out the rest using some sort of quality threshold? Or do we need do something more sophisticated?

The famous DBSCAN [6] algorithm partially solves this problem for Euclidean space by placing a fixed-radius spherical ball at each data point, and then counting the number of points falling within this spherical neighborhood, and considering that to be proportional to the density at that point. It then prunes all points that don't have enough neighbors (passed as a parameter) before performing clustering. This enables robust filtering of the small noisy clusters. However, DBSCAN cannot discover dense clusters at multiple resolutions/densities as it uses a single threshold for density;

*www.lightsphereai.com

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
MLG2020, August 24, 2020, San Diego, California
© 2020 Association for Computing Machinery.
ACM ISBN TBD.
https://doi.org/TBD

while 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 [10], and further unified with other related algorithms by [4]. [4, 10] are based on the concept of Hierarchical Mode Analysis (HMA), first proposed in [25], 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 [10]. 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.

Approaches such as Single-Link Agglomerative Clustering (SL) [8] do apply to the clustering of objects specified via pairwise similarity values, and give a range of clusters from N down to one. By stopping at an intermediate stage in the agglomeration process, and discarding singletons, one can obtain clusters over part of the data while ignoring some objects that do not cluster well at that stage. However, these approaches do not do so well in complex, noisy settings that we focus on in this paper, where they tend to find many small noisy clusters.

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 two real-world problems that we encountered. To concretize the issues being addressed, we now describe these two applications, for which we are also open-sourcing corresponding datasets.

1.1 Two motivating, real-world applications

Real-time Social Viral Marketing: The first 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). A post can belong to either a potential consumer or an influencer. 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 shared interests, to whom we may want to send highly targeted messaging campaigns, or belong to active influencers to whom we want to reach out to help with social viral marketing.

The desired clusters of posts tend to naturally form either around single influencers with a large number of highly related marketing posts, or over a larger number of consumers with a smaller number of more organic posts. Influencer and consumer groups naturally exist at varying degrees of "density" or cohesiveness depending on the type of consumers or influencers, and 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.

Topic Categorization: The second application relates to an Education product from Lighsphere AI, where there was a need for categorizing topics discovered from an upstream topic modeling system, for a relevant sub-area (Databases or Machine Learning) within a discipline (Computer Science). The data consists of a set extracted papers with associated co-authors and topics. The clustering of the topics is an input to a downstream recommender application that recommends educational topics to students and professionals for self-learning and career advancement advice. Each topic is a node in the graph, where two topics are connected by a weighted similarity edge if the similarity between two topics is above a certain threshold. The similarity depends upon the degree of co-occurrence of the two topics with the same set of authors, papers, and other topics.

The (sub)-topics at the highest resolution are numerous, and tend to be noisy, with a large fraction of the topics not forming any relevant categories or being irrelevant for the recommender application. Also, natural topic clusters can form at multiple resolutions based on the type of the cluster, which can either be a collection of synonyms describing what is semantically one topic, or represent a higher level category or a "meta-topic". Synonym clusters tend to be of higher "density" while the meta-topic clusters tend to be broader. For these reasons, finding all the topic clusters at multiple resolutions while pruning out quite a significant fraction of noisy topics was imperative for this problem.

1.2 Outline

To begin addressing the challenging settings described above, we start by first defining a flow-based [12] notion of density for nodes in a graph that we call flow density (Section 3.2). The resolution or scale of flow density can be controlled by a similarity threshold, and is analogous to the radius of spheres used in spatial HMA that controls resolution and can prune out less dense regions. Then, we present a fast algorithm that can incrementally grow the most dense regions of the graph by varying this similarity threshold to discover sub-graphs at variable densities/resolutions. We call our algorithm HIMAG, which is presented in Section 4.

In Section 5.1, we present an edge-based measurement formulation that allows us to evaluate clusterings of dense sub-graphs, including two new metrics: Edge ARI and Point Precision, both specially designed to work with edge-based partially labeled data. We are also open-sourcing associated tools that allow practitioners to set up these measurements easily for new datasets and algorithms for benchmarking and development purposes.

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 seven of the datasets studied in Section 5. To help one see the results, we also introduce a multi-resolution clustering visualization tool that works with graphs, Gene DIVER 3.0, which is being released with this paper. It is a software tool extended on top of the Gene DIVER 2.0 which was originally released with [10]. It provides a user interface to browse the cluster hierarchy from HIMAG.

2 RELATED WORK

2.1 Spatial HMA Methods

In Section 1, we described briefly how [4, 10] use a three step process to identify multi-resolution dense clusters on spatial data. [1, 4, 6, 10, 17] are all related to HMA [25] proposed in 1968, and fall into a class of non-parametric density-based clustering algorithms based on the idea that, given a set of i.i.d. data points from a multivariate distribution in a metric space, the observed "modes" or dense regions of the data set need not satisfy any particular parametric shape.

2.2 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 [16], KaHIP [19, 20], KaHyPar [13, 21], and PaToH [27]. 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 (see Section 3.2) partitioning and also supports parallelism [19]. KaHyPar is also a multilevel hypergraph partitioning algorithm utilizing an excess incremental breadth-first search (IBFS) maximum flow algorithm [7] and Boykov–Kolmogorov max-flow algorithm [2]. PaToH uses greedy hypergraph growing [26] and Boundary Fiduccia–Mattheyses algorithms [24]. 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. 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 5) show that even with these enhancements, the algorithms still do not perform as well as the approach introduced in this paper.

3 PRELIMINARIES

3.1 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. $\{a, b, c\}$), or with a predicate (e.g. $\{x \in \mathbb{R} | x < 7\}$). $\mathbb{Z}^+$ represents the domain of positive integers. Bold-faced capital letters such as $\mathbf{M}_D$ represent a two-dimensional matrix, which is accessed as $\mathbf{M}_D(i, j)$ (for row i, column j; both 1-indexed). A tuple is denoted by parentheses (e.g. (a, b, c)), and can be of any length.

3.2 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 [11, 12], is given by:

$$flow_\epsilon(a, b) = \sum_{i=1}^{n} \mathbf{S}_\epsilon(a, i)\mathbf{S}_\epsilon(i, b) \tag{1}$$

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:

$$\rho_{flow_\epsilon}(i) = \sum_{j=1}^{n} flow_\epsilon(i, j) \tag{2}$$

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 4.

3.3 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 [9], connectivity [11], and shared neighbor similarity [5] have also been used. These flow-based methods [5, 9, 11] 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 2 and 5). The reason for the improvement in performance 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 sub-graphs 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 define dense regions in the graph to find dense clusters. This dilemma is resolved by the notion of flow density (Equation 2), 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 the flow density of nodes, 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 nodes that are not tightly connected to a dominates, creating a smoothing effect on the measure of node density. In contrast, when s_ϵ is large, only a few of the nodes are connected to each other and the flow density of a is only contributed to by nodes that have a large s_ϵ to a, resulting in flow density at a node only being affected by "nearby" points in the similarity graph. Thus s_ϵ, when used with flow density, controls the resolution and smoothness of density variations in the graph visible to the clustering algorithm, and is exploited by our HIMAG algorithm 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.

Table 1: 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
Databases	Topic	5,699	14,115,993
Machine Learning	Topic	7,833	11,361,558
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+

3.4 Benchmark Graphs & Datasets

We now discuss in brief the construction of graphs from various datasets used in this paper. These graphs are the input to our algorithm and to the other graph methods we compare with.

Sim-2 Graph for deep analysis: [10] uses a synthetic 2-D dataset sampled from 5 spherical Gaussians, and an additional background distribution as their synthetic / development dataset. Such a dataset can be very helpful when developing and analyzing complex dense cluster discovery algorithms, since the true clusters and background noise points are perfectly known and visually comprehensible. It turns out that there is a simple transform that can be applied to the Sim-2 data to produce a graph that we call Sim-2 Graph, which we use for further analysis into our method. The transformation computes similarity between points i and j as $\mathbf{S}(i,j) = 1 - \frac{\mathbf{D}_S(i,j)}{\max \mathbf{D}_S}$, where $\mathbf{D}_S$ is the euclidean distance matrix calculated from the original data. This graph $\mathbf{S}$ can then also be pre-thresholded, before being passed to the algorithms.

New Applications: Topic & Marketing Graphs: Graphs stemming from the two applications mentioned in the introduction were constructed by using the Jaccard metric [18], which is ideally suited for constructing similarity graphs that are in a non-metric space, especially when the terms are sparse [22]. For the marketing datasets, the similarity is measured by shared hashtags between posts, which are the nodes of the graphs. For topic graphs, the similarity between two topics is measured by shared co-occurrence of the two topics within the same paper, or being used by the same author in their papers, or both co-occurring in a paper with the same third topic. If we treat the shared authors or papers or topics as the shared terms, and for the marketing graph posts nodes if we treat the shared hashtags as shared terms, then the Jaccard is computed simply as the count of shared or intersecting terms between two nodes divided by the union of all the terms occurring in either node, and is a value between 0 and 1.

Standard social network benchmarks: We also present results on two standard social network datasets: Pokec [23], which is Slovakia's largest social network, and the LiveJournal social network [15]. These two datasets are valuable as they are large, fairly well-known, and open-sourced social graph datasets. We were able to utilize them to set up a social network prediction problem by showing only a small part of the graph to the graph algorithms to predict the dense sub-graphs, and then use the remaining graph as "labels" for predicting "new" connections/friendships between people not present in the training graph. The accuracy of the predictions was then used to compare the different methods. This is a valuable benchmark setup as it allows us to compare our method against a broad set of algorithms and parameters, without the need for expensive human labeling for each result. This is in contrast with our new real-world applications for Topic and Marketing datasets where the clustering results from various methods had to be evaluated manually by independent human labels, which were tedious and expensive to collect.

Table 1 summarizes the graphs generated from these datasets, and some more details on parameters for the graph construction are discussed further in Section 5.2.

4 HIMAG

Our new *Hierarchical Incremental Mode Analysis for Graphs* algorithm outputs an HMA matrix like the one described in [10]. We then identify clusters across resolutions and relabel them, and the rows (nodes) are then sorted using a dictionary sort for better visualization. Clusters are then each assigned a stability using the algorithm presented in [10], which has been reproduced in Equation 3.

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.

4.1 Graph HMA

HIMAG works by redefining the key notions of spatial radius (r_ϵ), spatial density (ρ_ϵ), and density threshold (n_ϵ) used in the Spatial HMA algorithms [4, 10, 17]. As described in Section 3.2, 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 $\mathbf{S}_\epsilon$ 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, which is associated with a specific threshold s_ϵ. The values in the matrix represent cluster IDs, which are generated independently for each level.

The list of values to use for s_ϵ are generated based on the input similarity matrix $\mathbf{S}$. [10] calculates the 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 $\mathbf{S}_\epsilon$ (given by $|\{\mathbf{S}_\epsilon(i,j) | i,j \in \mathbb{Z}^+ \wedge i < n \wedge j < n\}|$) between two successive levels remains a constant r_{shave}.

4.2 HIMAG Algorithm

The HIMAG algorithm provides a computationally efficient way to compute the Graph HMA matrix described in Section 4.1.

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

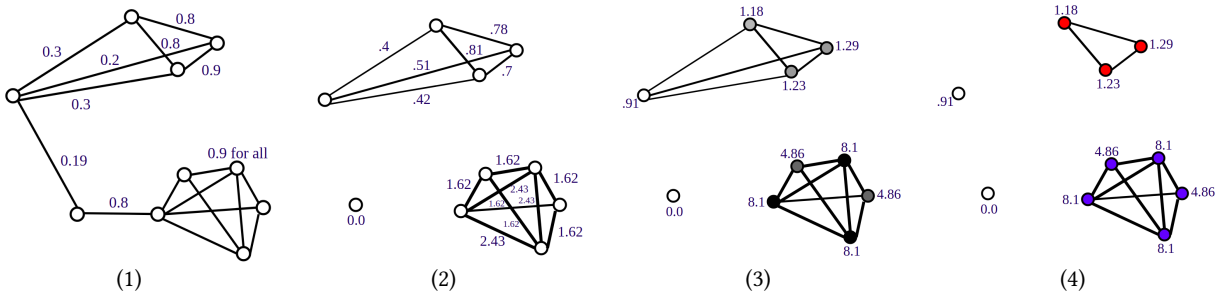


Figure 1: A simplified illustration of one level of HIMAG Graph clustering showing (1) a sparse undirected weighted graph, (2) kept all edges for $s_\epsilon > 0.2$, compute edge flow, (3) compute node density ρ_{flow} and prune non-dense points with $\rho_{flow} < n_{flow_\epsilon}$, (4) cluster all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

with weight s. This sparse representation saves memory and allows us to perform some optimizations.

First, we sort all the edges in the input graph $\mathbf{S}$, and store that in a list $\mathbf{l_s}$. This allows us to easily compute the list of values of s_ϵ to be used, $\mathbf{s_t}$, such that the fraction of edges with weights between any two successive values in $\mathbf{s_t}$ is a constant r_{shave}. We can also ensure $\mathbf{s_t}$ contains no duplicates. Next, we partition $\mathbf{l_s}$ into buckets of edges with similarities between successive thresholds in $\mathbf{s_t}$. 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 $\mathbf{S}_\epsilon$, as 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 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 now 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 $\mathbf{S}_\epsilon(i,a) * \mathbf{S}_\epsilon(a,b)$, where $\mathbf{S}_\epsilon(a,b)$ is given by s. Similarly, $\forall i \in \mathbf{e_{nbr}}(b)$, we increment $\rho_{flow_\epsilon}(i)$ and $\rho_{flow_\epsilon}(a)$ by $\mathbf{S}_\epsilon(i,b) * s$. And finally we add node a to $\mathbf{e_{nbr}}(b)$ and node b to $\mathbf{e_{nbr}}(a)$.

As we add edges, we also keep track of which nodes' flow density increased that were not already clustered. After adding all the edges in one bucket (i.e. all the edges above one of the thresholds s_ϵ have been accounted for), 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 C according to the connected components rules, merging clusters when they get connected, or creating new ones as necessary.

After this, the clustering C at the current level k is saved into the HMA matrix $\mathbf{L_{HMA}}$. Each of the clusters in C is assigned a positive integer ID, and then each node i 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 process grows 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. $\mathbf{L_{HMA}}$ can then have its leading columns pruned, and this further saves computation time. This is quite useful for many practical applications where the denser subset of the topology discovered is more important.

4.3 HMA Visualization & Cluster Selection

Since the HMA matrix from HIMAG has an identical form to spatial Auto-HDS, we follow the same process as [10] for visualization. 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 C_i first comes into existence and when it disappears, in the order from left to right (shrinking clusters) in the $\mathbf{L_{HMA}}$ matrix, then we define *stability* as:

$$Stab(C_i) = \frac{\log(f_C(start_i)) - \log(f_C(end_i))}{\log(0.99)} \tag{3}$$

This notion of stability above has one improvement over [10] in that the denominator in Equation 3 is set to $\log(0.99)$, corresponding to $r_{shave} = 1\%$ (arbitrary normalization) instead of the variable $\log(1 - r_{shave})$. As noted in [3], this makes the notion of stability independent of the shaving rate, thus making it possible to get the log-scale stability irrespective of the whether the algorithm computes the full HMA hierarchy or a logarithmic pruned one. Note that it is trivial to modify HIMAG to simply compute all the levels; because of the incremental nature of the algorithm, the computational overhead is not significant. However, the storage of the HMA matrix becomes $O(n)$ instead of $O(\log(n))$, for only a small gain in cluster resolution.

5 EXPERIMENTAL EVALUATION

In this section, we first define two new evaluation metrics suitable for sparse graphs. Then, we describe the experimental setup for the seven datasets summarized in Table 1, and present results in Table 2, Table 3, and Figure 2.

5.1 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 [14] that was designed for partitional clustering with edge-based labels.

Table 2: Point Precision performance comparison for Topic and Marketing Segments comparing HIMAG with FLOW (F) variation of hMETIS for the Top 100 or 50 (by stability) and All judged clusters. The Runtimes (RT) of the algorithms are shown in seconds on an 8-core Lenovo ThinkServer machine with 32 GB RAM. HIMAG recall for All clusters for Machine Learning (ML) was 0.140 with $k = 221$ clusters, while for Database (DB) the recall was 0.163 with $k = 178$ 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	ML				DB				Wellness				Education			
	Top 100	All	S	B	Top 100	All	S	B	Top 50	All	S	B	Top 50	All	S	B
HIMAG	.9088	.9107	2	22	.8227	.8479	2	46	.9394	.9438	7	1388	.9973	.9748	2	248
hMETIS 2k, F	.8735	.4886	2	6	.4530	.2543	2	24	.7665	.718	7	159	.8953	.7336	2	87

Given a clustering C containing clusters 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 $\mathcal{L}_S$ containing all pairs (a, b) that are linked ("must-link"), we define $C_p = \{(a, b) | \exists C_i \in C; (a, b) \in C_i\}$ as the set of all edges predicted as linked in a clustering C. We then define the *Edge Precision* metric as:

$$P_E(C, \mathcal{L}_S) = \frac{\left|C_p \cap \mathcal{L}_S\right|}{\left|C_p\right|} \tag{4}$$

If all the must-link edges found within a cluster 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 $(C_i \cap \mathcal{L}_S)$, this hypothetical c can be calculated with $c = \frac{1}{2}\left(1 + \sqrt{8\left|C_i \cap \mathcal{L}_S\right| + 1}\right)$, 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 follows:

$$P_p(C, \mathcal{L}_S) = \sum_{C_i \in C} \frac{\frac{1}{2}\left(1 + \sqrt{8\left|C_i \cap \mathcal{L}_S\right| + 1}\right)}{|C_i|} \tag{5}$$

For computing *Edge ARI*, an exhaustive label set is also required. Any edge not in $\mathcal{L}_S$ 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 [14]. The index is simply the number of edges that are "must-link" and are predicted in the clustering 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 C, as both of these would have to be true for the edge to count towards the index. This gives us the following formula, where n is the total number of nodes in the labeled graph:

$$ARI_E(C, \mathcal{L}_S) = \frac{\left|C_p \cap \mathcal{L}_S\right| - \left|C_p\right| \frac{|\mathcal{L}_S|}{\binom{n}{2}}}{\min\left(\left|C_p\right|, |\mathcal{L}_S|\right) - \left|C_p\right| \frac{|\mathcal{L}_S|}{\binom{n}{2}}} \tag{6}$$

5.2 Experimental Setup for Comparison

Section 2.2 describes the graph algorithms that we adapt and compare with our method. The following enumerates some of details for the experimental setup used to benchmark our methods against those algorithms on various datasets:

Parameters Setup for Comparisons

(1) Exhaustive cannot-link labels are needed to compute Edge ARI correctly. Since our Topic and Marketing datasets were judged manually, Edge ARI was not possible to compute, so only Point Precision results are presented on them.
(2) To plot the performance by fraction of data clustered, we took the most stable non-overlapping clusters from HIMAG. This gives us not only the total fraction of data clustered, but also allows us to plot increasing fraction of data clustered by including progressively less stable clusters.
(3) 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 some cases where they performed poorly compared to HIMAG when the k was the same.
(4) The results of all of the methods we compete against were much worse on the Topic and Marketing data for **k** than for **2k**, and the labeling was expensive, so we only present results for **2k** on those datasets.
(5) Since HIMAG clusters only a fraction of data, two different methods were used to shave the clusters obtained from the partitional algorithms to produce smaller clusters and to have the same fraction of data clustered as HIMAG. **RAND** represents randomly selecting a fraction of the points from each cluster, while **FLOW** represents shaving each cluster by removing the least dense nodes until the desired number of nodes remain.

All methods received exactly the same input graph for each dataset. More details, along with the source code for all algorithms and measurements will be provided with a supplement link to ensure these experiments are fully reproducible, but here are some important details for the construction of graphs:

Other parameters for graph construction:

(1) A regularization term was added to the denominator of the Jaccard before computing the ratio, as otherwise, nodes with a small number of terms generate noisy high similarity edges.

This substantially improves results on these datasets for all algorithms.

(2) We also pre-thresholded most graphs for speed, as the low similarity edges don't capture any additional information for finding high quality dense sub-graphs. A threshold of 0.25 was used to prune edges on Marketing data, and 0.01 for Topic.
(3) For topic data the Jaccard similarity was weighted using the formula described in [11].
(4) For the standard social graphs, the "interests" were the terms used for computing the Jaccard similarities between two people. We use the community profile that these two datasets provide, namely "hobbies" for Pokec and "communities" for LiveJournal, to compute this interests graph. Only 2% of the edges were used for building the graphs, while the remaining 98% of the graph edges were held back and used as "labels".

6 GENE DIVER VISUALIZATION & TOOLS

We developed Gene DIVER 3.0 for our experimental evaluation, as an extension to Gene DIVER 2.0 released in [10]. 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].

7 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.

ACKNOWLEDGMENTS

This research was supported by Lightsphere AI. We thank Andrew Galloway and Mary Remya for helping us build the machinery for marketing and topic modeling data. We are also grateful for the countless hours spent by Alex Tarasar and his judging team towards organizing the pipeline that produced labels and analysis for these datasets. His team included Ashley Prietto, Sandrine Gupta, and Kevin McCrea.

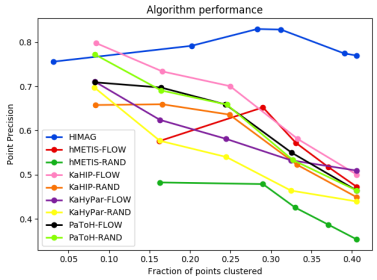
[1] github.com/nog642/himag-release

REFERENCES

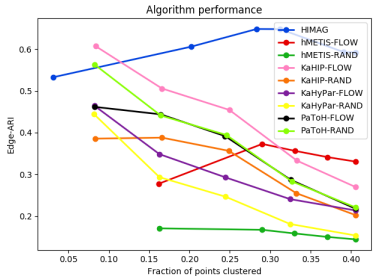
[1] M. Ankerst, M. Breunig, H. P. Kriegel, and J. Sander. 1999. OPTICS: Ordering Points To Identify the Clustering Structure. In *ACM SIGMOD Conference*.
[2] Yuri Boykov and Vladimir Kolmogorov. 2004. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26, 9 (September 2004), 1124–1137.
[3] Ricardo J.G.B. Campello, Davoud Moulavi, and Jörg Sander. 2012. A Simpler and More Accurate AUTO-HDS Framework for Clustering and Visualization of Biological Data. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 9, 6 (August 2012), 1850–1852.
[4] Ricardo J. G. B. Campello, Davoud Moulavi, Arthur Zimek, and Jorg Sander. 2015. Hierarchical density estimates for data clustering, visualization, and outlier detection. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 10 (July 2015), 5. Issue 1.
[5] Levent Ertöz, Michael Steinbach, and Vipin Kumar. 2003. Finding Clusters of Different Sizes, Shapes, and Densities in Noisy, High Dimensional Data. 47–59.
[6] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. 1996. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *In Proc. KDD-96*.
[7] Andrew Goldberg, Sagi Hed, Haim Kaplan, Pushmeet Kohli, Robert Tarjan, and Renato Werneck. 2015. Faster and More Dynamic Maximum Flow by Incremental Breadth-First Search. In *Algorithms - ESA 2015*, Vol. 9294. Sorubger, 619–630.
[8] J. C. Gower and G. J. S. Ross. [n. d.]. ([n. d.]).
[9] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. 1999. ROCK: A Robust Clustering Algorithm for Categorical Attributes. In *15th International Conference on Data Engineering*. Sydney, Australia, 512.
[10] Gunjan Gupta, Alex Liu, and Joydeep Ghosh. 2008. Automated Hierarchical Density Shaving: A robust, automated clustering and visualization framework for large biological datasets. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 4, 7 (2008).
[11] Gunjan K. Gupta and Joydeep Ghosh. 2001. Value Balanced Agglomerative Connectivity Clustering. In *SPIE conference on Data Mining and Knowledge Discovery III, SPIE Proc.*, Vol. 4384. Orlando, Florida, 6–15.
[12] George T. Heineman, Gary Pollice, and Stanley Selkow. 2008. *Network Flow Algorithms.* Oreilly Media, Chapter Chapter 8, 226–250.
[13] Tobias Heuer, Peter Sanders, and Sebastian Schlag. 2018. Network Flow-Based Refinement for Multilevel Hypergraph Partitioning. In *17th International Symposium on Experimental Algorithms (SEA 2018)*. 1:1–1:20.
[14] L. Hubert and P. Arabie. 1985. Comparing partitions. *Journal of classification* (1985), 193–218.
[15] A. Dasgupta M. Mahoney. J. Leskovec, K. Lang. 2009. Community Structure in Large Networks: Natural Cluster Sizes and the Absence of Large Well-Defined Clusters. *Internet Mathematics* 6, 1, 29–123.
[16] George Karypis, Rajat Aggarwal, Vipin Kumar, and Shashi Shekhar. 1997. Multilevel hypergraph partitioning: Applications in VLSI domain. In *Design and Automation Conference*.
[17] Leland McInnes and John Healy. 2017. Accelerated Hierarchical Density Based Clustering. In *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*. IEEE, 33–42.
[18] G. Salton. 1989. *Automatic text processing: the transformation.* Addison Wesley.
[19] Peter Sanders and Christian Schulz. [n. d.]. KaHIP v0.6 – Karlsruhe High Quality Partitioning. ([n. d.]).
[20] Peter Sanders and Christian Schulz. 2013. Think Locally, Act Globally: Highly Balanced Graph Partitioning. In *Proceedings of the 12th International Symposium on Experimental Algorithms (SEA'13)*, Vol. 7933. Springer, 164–175.
[21] Sebastian Schlag, Christian Schulz, Daniel Seemaier, and Darren Strash. 2019, to appear.. Scalable Edge Partitioning, In Proceedings of the 21th Workshop on Algorithm Engineering and Experimentation (ALENEX). *Technical Report, arXiv:1808.06411*.
[22] Alexander Strehl, Joydeep Ghosh, and Raymond J. Mooney. 2000. Impact of Similarity Measures on Web-page Clustering. In *AAAI Workshop on AI for Web Search (AAAI 2000)*. AAAI/MIT Press, 58–64.
[23] Lubos Takac and Michal Zabovsky. 2012. Data analysis in public social networks. *International Scientific Conference and International Workshop Present Day Trends of Innovations* (May 2012), 1–6.
[24] Vadim Olshevskyand Eugene Tyrtyshnikov. 2010. *Matrix Methods: Theory, Algorithms And Applications - Dedicated To The Memory Of Gene Golub.* World Scientific Publishing Company.
[25] D. Wishart. 1968. Mode Analysis: A generalization of nearest neighbour which reduces chaining effects. In *Proceedings of the Colloquium in Numerical Taxonomy*. Academic Press, University of St. Andrews, Fife, Scotland, 282–308.
[26] Ümit V. Çatalyürek and Cevdet Aykanat. 1999. Hypergraph-partitioning-based decomposition for parallel sparse-matrix vector multiplication. *IEEE Transactions on Parallel and Distributed Systems* 10, 7 (July 1999), 673–693.
[27] Ümit V. Çatalyürek and Cevdet Aykanat. 1999. PaToH: Partitioning Tool for Hypergraphs. (November 1999), 33 pages.

Table 3: 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.

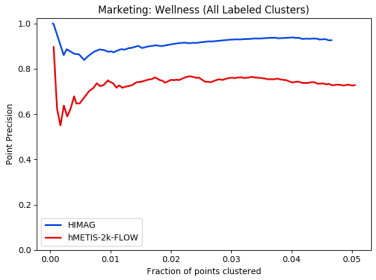
Algorithm	Pokec						LiveJournal						Sim-2					
	PP	E-ARI	RT	S	B	C	PP	E-ARI	RT	S	B	C	PP	E-ARI	RT	S	B	C
HIMAG	.121	.0214	147.88	3	74	8769	.201	.2541	343.18	3	488	6002	.770	.5919	68.4	20	223	6
hMETIS 2k, F	.031	.0102	58.34	2	13	17538	.055	.0229	77.168	2	71	11994	.479	.3107	52.9	36	76	10
KaHIP 2k, F	.026	.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



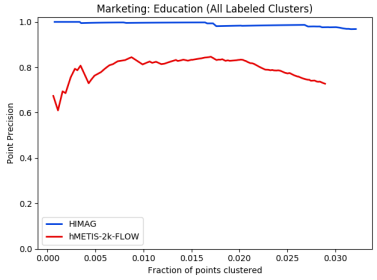
(Sim-2 Point Precision)



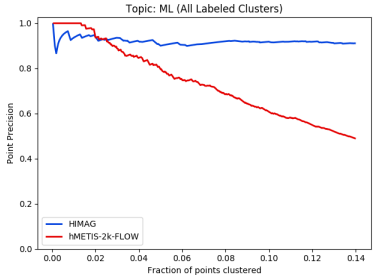
(Sim-2 Edge ARI)



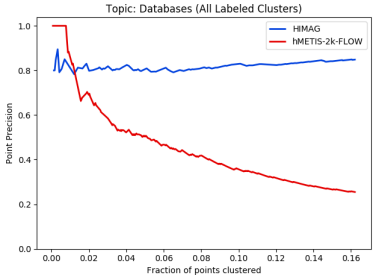
(Wellness Point Precision)



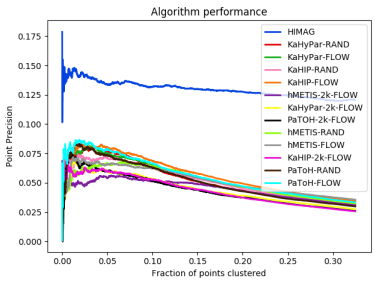
(Education Point Precision)



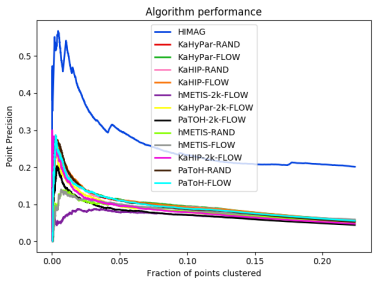
(ML Point Precision)



(DB Point Precision)



(Pokec Point Precision)



(LiveJournal Point Precision)

Figure 2: Edge ARI and Point Precision plots for Sim-2, Marketing, Topic, 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.