Local Clustering over Labeled Graphs: An Index-Free Approach

Niu Yudong, Yuchen Li
Singapore Management University
Singapore
ydniu.2018@phdcs.smu.edu.sg, yuchenli@smu.edu.sg

Ju Fan
Renmin University of China
Beijing, China
fanj@ruc.edu.cn

Zhifeng Bao
RMIT University
Melbourne, Australia
zhifeng.bao@rmit.edu.au

***Abstract*—In this paper, we study local clustering over labeled graphs, which extracts a subgraph with nodes having high label density matched to the query labels as well as high structure density around a seed node. Despite the progress made in the last few years, we observe two major limitations of existing methods: (I) The candidate subgraphs have to comply with strict topology-driven models and better candidates can be pruned by these topological constraints; (II) The topological constraints give rise to substantial computational overheads and existing works have to construct prohibitively large indexes for online processing. To mitigate these limitations, we explore the idea of using conductance in local clustering that ensures structure density through minimizing conductance. Conductance is a well-understood metric primarily for detecting unlabeled clusters but for labeled graphs, applying conductance directly is insufficient because the label information is not taken into consideration. To this end, we propose a novel Label-Aware Motif weighted framework (LAM) to transform the labeled graph to a weighted graph so that both the label and the structure proximity of nodes are captured. We define label-aware motifs as small high-order structures of nodes with query labels. Nodes within a label-aware motif are both closely connected and relevant to query labels, which ease the process of identifying labeled clusters. Our theoretical study shows that LAM is able to better distinguish the desired candidates under the personalized pagerank distribution from the seed node on random graphs generated by the stochastic block model. Based on such nice properties of LAM, we propose an index-free peeling algorithm to efficiently search local clusters on labeled graphs. Extensive experiments on both real-world and synthetic networks show that our proposed algorithm can achieve up to 90% relative effectiveness improvements (F1 scores), while using 10 times less memory than the SOTA algorithm.**

I. Introduction

The graph model has emerged as a prevalent tool to enable analysis over growing complexity of big data. By modeling relationships between entities, such as persons, genes or transactions, insights can be extracted from the graph structure for numerous domains [1], [2]. Nevertheless, many real-world graphs are labeled graphs, which associate nodes with labels. For instance, in citation networks, papers are modeled as nodes and the node label captures the topic of the paper. Two nodes are connected by an edge if there is a citation relationship between the corresponding papers. In gene interaction networks, a set of genes (nodes) are connected by edges representing the functional correlation between two genes and each node is labeled with its corresponding gene's functional attributes. Node labels contain additional information about entities besides the graph structure and it is crucial to utilize these information properly in various analysis tasks to achieve better results.

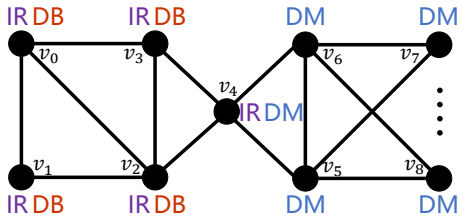


Fig. 1: A labeled graph G with a desired cluster $H^* = \{v_0, v_1, v_2, v_3\}$ given the seed node v_0 and query labels $\{\texttt{DB}, \texttt{IR}\}$. The small dots represent the remaining nodes in the graph

In this paper, we study the problem of local clustering on labeled graphs to support group-level analysis: given a set of query labels, we find a labeled cluster (i.e., a densely connected subgraph) around a seed node, while ensuring that the nodes in the discovered cluster have similar labels to the query labels. The set of query labels is used to find clusters with labels that are relevant to the user's interests.

Example 1. *Fig. 1 displays a small part of an academic collaboration graph G. Each node in G denotes a researcher and the labels associated with the node denote her research areas. Each edge represents a collaboration relationship between two researchers. Users can issue a local clustering query from a researcher as the seed node (e.g., v_0 in Fig. 1) and areas of interest (e.g., labels $\texttt{DB}, \texttt{IR}$) to discover a subgraph that is not only densely connected, but also relevant to the query labels, e.g., $H^* = \{v_0, v_1, v_2, v_3\}$.*

Applications. Local clustering on labeled graph can benefit various real-world applications. (1) Social marketing. Online advertisement through social networks has become an important approach to boosting sales. Label-aware local clustering on social networks can help advertisers choose advertising targets. For example, to place ads for certain instrument, the advertiser can find a cluster with label "music" around a user who has already bought this instrument and push ads to users in this cluster. The seed node user who has bought this instrument is important for effectiveness of the ad since it is more likely for users to trust the ad if they have heard

about it from their friends. (2) Protein function analysis. Label-aware local clustering can be applied on protein-protein interaction (PPI) networks to aid function analysis of proteins. Proteins that have similar functions will interact with each other frequently and thus form a cluster in the PPI network. To determine the function of a given protein, the researchers can launch queries with different function labels from the given protein. The protein will have a certain function with high probability if a good labeled cluster can be found around the protein. Such promising candidate functions will then be thoroughly studied by lab experiments [3], [4], [5].

Prior works. A line of study has focused on a similar problem called community search on labeled graphs. However, this line of works have two major drawbacks. First, most if not all existing works for community search on labeled graphs have to define communities through topology-driven models such as k-core [6] and (k, d)-truss [7], which employ topological constraints on the resulting community. Thus, some desirable clusters can be overlooked when they do not satisfy the strict topological constraints. As verified in our experiments, these methods based on topology-driven models only achieve inferior effectiveness to our proposed methods even if the model parameters are fine-tuned specifically for each query. Second, to enable online processing for large graphs, it requires prohibitive indexes to identify candidate communities satisfying the topological constraints. For large graphs such as `friendster` in our experiments, constructing the SOTA indexes [7] requires more than 256GB memory and overwhelm the memory size of the server used in our experiments. Thus, it is demanding to call for index-free methods that are scalable for detecting labeled clusters without strict topological constraints.

Our work. In this paper, we propose to search local clusters on labeled graphs with a *conductance-driven* approach. Conductance [8], [9], [10], as a well-understood metric for graph clustering on unlabeled graphs, measures the ratio between the number of out-going edges and the total number of edges of a candidate cluster. A small conductance implies a densely connected cluster as the number of edges going out of the cluster is relatively small compared with the number of edges within the cluster. The benefit of adopting conductance is two-fold: (1) conductance does *not* impose strict topological constraints and extends the search space of the candidate clusters compared with the community search methods; (2) efficient *index-free* algorithms are available to quickly identify dense clusters with small conductance values. Nevertheless, a naïve adoption of conductance leads to suboptimal quality on labeled graphs as conductance does not consider the synergy between label and structure information for labeled clusters.

Motivated by the above, we propose a novel and intuitive *Label-Aware Motif weighted* framework (LAM) to fuse structure and label information in the conductance model. LAM considers label-aware motifs, which are *triangle* instances where the labels of each node in the triangle have non-zero overlap with the query labels. The core idea of LAM is to transform a graph G into a weighted graph G_M by assigning a weight to each edge based on the number of query motifs containing that edge. Prior works for high-order graph clustering [11], [12], [13], [14] have shown the significance of utilizing motifs (especially triangles) in clustering to capture the structure proximity between nodes. The LAM model further captures the label and the structure proximity between any two nodes simultaneously in edge weights through considering label-aware motifs. Then, the edge weighted graphs are fed to the weighted conductance model for detecting labeled clusters. Notably, unlike the indexes for topology-driven models which require time-consuming offline preprocessing of the whole graph G, our method only requires constructing G_M *online* and *locally* around the seed node, which is both time and space efficient.

We theoretically show that, with the LAM framework, the PPR (Personalized PageRank) distribution will become more concentrated within the labeled cluster around the seed node on random graphs generated by the stochastic block model. The concentrated PPR distribution will benefit conductance minimization algorithms [9], [15], [16] – nodes in the desired cluster are boosted with higher PPR values so that they are easily distinguished from nodes outside the cluster. Our analysis is also applicable to other random-walk ranking models such as the *heat-kernel* [16] and the *inversed* [17] pagerank.

On top of the LAM framework, we further propose an efficient *index-free* algorithm to address local clustering on labeled graphs. Since conductance optimization is NP-hard [18], we propose a two-stage heuristic algorithm for efficient cluster detection. In the first stage, we employ the general sweep algorithm [9], [15], [16] to efficiently extract a cluster as a coarse-grained candidate H_0. Since some outliers may be included in this coarse-grained candidate, we further execute a peeling procedure in the second stage that iteratively trims unpromising nodes in H_0 to form smaller candidate clusters. Finally, we identify the best cluster H^* among the candidates. We conduct extensive experiments on both real and synthetic datasets. The experimental results show that our proposed method not only recovers the ground-truth clusters significantly better than the topology-driven methods, but also achieves superior performance on time and memory efficiency.

The contributions of this work are summarized as follows:

- We propose a novel LAM framework and prove that LAM can effectively concentrate the PPR value within the labeled cluster around the seed node. To the best of our knowledge, this is the first work for local clustering driven by the concept of motifs on labeled graphs (Sec. III).
- We devise a two-stage algorithm based on the LAM framework. One highlight of our algorithm is *index-free*, which makes it possible to process graphs with billions of edges. For a real network `friendster` with more than 66 million nodes and 1.8 billion edges, the existing topology-driven method fails to construct the indexes due to prohibitive memory consumption. In contrast, our method well supports `friendster` without resorting to costly indexes (Sec. IV).

- We conduct extensive experiments on both real and synthetic networks to validate the effectiveness as well as the efficiency of the proposed approach over the state-of-the-art methods. LAM consistently outperforms the baselines and achieves up to 90% relative improvement on F1 scores. The two-stage algorithm shows similar or better efficiency while using 10x fewer memory than the indexed-based approaches (Sec. V).

II. Preliminaries & Literature Review

In this section, we first introduce the basic notations and problem formulations of local clustering on labeled graphs. Subsequently, we discuss the related literature.

A. Notations and Problem Formulation

Notations. In this work, we consider an undirected *labeled* graph $G = (V, E, \mathbb{L})$ with node set V, edge set E and label set $\mathbb{L}$. Each edge $e = (u, v) \in E$ connects two nodes u and v, and each node u is associated with a set of labels, denoted by $L(u) \subseteq \mathbb{L}$. For ease of presentation, we denote the set of neighbors of a node $u \in V$ by $\mathcal{N}(u)$, and the degree of u by $d(u) = |\mathcal{N}(u)|$. For the given query label set $L_q \subseteq \mathbb{L}$, we use $V(L_q)$ to denote the set of nodes associated with at least one label within L_q, i.e. $V(L_q) = \{u | L(u) \cap L_q \neq \emptyset\}$. Without loss of generality, we consider that graph G is connected.

Local Clustering on Labeled Graphs. A fundamental analytical task is to find clusters. For labeled graphs, users can input query labels to find dedicated clusters that are relevant to the queries, i.e., nodes in the cluster should contain relevant labels specified by users. Furthermore, global clustering algorithm is often prohibitively time-consuming for large graphs [19]. Following existing works [9], [15], [11], we study local clustering on labeled graphs where the clusters are extracted around any seed node.

It is worth highlighting that how to select the seed node is orthogonal to this work. Interested users can refer to existing works on how to pick a good seed node [10], [20], [21]. In this work, we follow [7] and randomly select the seed node from each ground-truth cluster to avoid the influence of different seeding algorithms.

Definition 1. *Given a labeled graph $G = (V, E, \mathbb{L})$ and a set of query labels $L_q \subseteq \mathbb{L}$, we aim to find a cluster H^* to maximize the metric $f_q(H)$ among all clusters containing a seed node s,*

$$H^* = \underset{\{H \subset V | s \in H\}}{\arg\max} \; f_q(H)$$

where $f_q(H)$ is a cluster metric that simultaneously measures the query label density and structure density of H.

The objective function $f_q(H)$ can be defined as the product of two sub-functions, representing the structure density and query label density of H respectively.

Example 2. *Existing works define the structure density as an indicator function of certain topological model. For example, the authors in [7] define the structure density of a subgraph H as $\mathbb{I}_{k,d}(H)$, where $\mathbb{I}_{k,d}(H) = 1$ if H is a (k, d)-truss, and $\mathbb{I}_{k,d}(H) = 0$ otherwise. The query label density $\rho(H, L_q)$ can be intuitively defined as the average number of query labels in H. Combining $\mathbb{I}_{k,d}(H)$ and $\rho(H, L_q)$, we can get a definition for objective function as $f_q(H) = \mathbb{I}_{k,d}(H) \cdot \rho(H, L_q)$. In Fig. 1, if we take $k = 1$, $d = 2$ and query labels as $L_q = \{DB, IR\}$, the score of $H^* = \{v_0, v_1, v_2, v_3\}$ is $f_q(H^*) = 1 \cdot \frac{8}{4} = 2$.*

Various $f_q(H)$ have been proposed by existing works and we review them in the remaining part of this section.

B. Related Work

We review closely related studies of community discovering on labeled graphs: (1) community search on labeled graphs; (2) global graph clustering on labeled graphs. The comparison between these related studies and this paper is summarized in Table I. Note that there are a plethora of works on local graph clustering or community search for unlabeled graphs [15], [9], [11], which are not the focus of our work. We refer the interested readers to the survey [22].

Community Search on Labeled Graphs aims at finding the community according to the query given by the user. Several community metrics $f_q(H)$ are proposed to consider structure and label densities simultaneously. According to different query inputs, the existing community search methods on labeled graphs can be further divided into three categories.

The first category, namely node-centric keyword-aware community search (NKCS), receives same inputs as our problem with both a set of query labels and a set of query nodes [6], [7], and aims to find a community that contains the query nodes and has high query label density within the community. Although methods in this category can be adopted to solve our label-aware local clustering problem to some extent, they adopt strict topological community models, like k-core [6], (k, d)-truss [7], and triangle-connected k-truss [23], and define the structure density in $f_q(H)$ as the indicator function of the model. Thus, these methods can overlook desirable clusters when they do not satisfy the rigid topological constraints. Besides, as discussed in Sec. I, these methods requires prohibitively large indexes for online processing and thus can't scale to large graphs[1]. In contrast, our proposed method finds local clusters efficiently without imposing hard constraints and index construction.

The second category, known as keyword community search (KCS), receives a set of query labels as inputs [24], [25], [26]. The target is to find a community in the labeled graphs such that the nodes in the community are most relevant to the query labels. Methods in this category cannot take the seed node as input. Thus, they are not customized to different users and often require the expensive global access of the entire graph. The third category [27], [28], namely node-centric community search (NCS), only takes a set of query nodes as the input and do not support query labels. VAC [27] is topology-driven

[1]There are two basic algoithms that do not require indexes for ACQ in [6]. However, these two methods are slower than index-based algorithms by several orders of magnitude and thus are not scalable to large graphs.

TABLE I: A comparison of representative community search works on labeled graphs.

Method	Seed Node	Query Labels	Topological constraints free	Index free
[24], [26]	✘	✔	✘	✘
[25]	✘	✔	✔	✔
[27]	✔	✘	✘	✔
[28]	✔	✘	✔	✔
[6], [7], [23]	✔	✔	✘	✘
[29]	✔	✘	✘	✘
Ours	✔	✔	✔	✔

and finds a k-truss formed by nodes with similar labels. [28] finds communities through GNN. However, the GNN requires manual supervision for finetuning the community. Furthermore, both methods in the third category do not find different communities according to query labels.

Recently, meta-path based community search over heterogeneous networks (HCS) was proposed in [29]. Instead of receiving query labels as the input, this work requires a query meta-path to strengthen the connections between nodes with the same type. However, its method cannot be used to solve our problem even if we regard labels on node as node types. The key difficulty lies in finding a meta-path that represents the set of query labels. The meta-path will inevitably impose structural constraints when strengthening the connection between nodes of same type, whereas the set of query label does not assume any structural constraints.

Global Graph Clustering on Labeled Graphs, also known as labeled community detection (LCD), divides a labeled graph into a number of partitions such that each partition is densely connected and the nodes in each partition share similar labels [30], [31], [32], [33], [34]. These LCD methods are not applicable to the problem studied in this work because: (1) The LCD methods always produce the same partition result and do not consider user queries; (2) The LCD methods employ global algorithms which are not scalable to large graphs. In this work, we allow users to input query labels for searching customized clusters and devise local algorithms to handle large graphs.

III. The LAM Framework

In this section, we present a Label-Aware Motif weighted framework (LAM) which fuses the structure information and the label information. The idea is to transform the original labeled graph G into a weighted graph G_M based on the query labels so that the desired clusters are easier to be extracted, regardless of the cluster metric used. Subsequently, we theoretically show that the framework concentrates the PPR distribution of the desired clusters for any seed node in that cluster. We focus on formalizing the LAM model in this section, and our index-free algorithm based on local LAM transformation will be introduced in Sec. IV.

A. Transformation under LAM

Motivated by the motif-aware graph clustering on unlabeled graphs [11], [12], [13], [14], we adjust the edge weights based on the number of *query-aware* motif instances containing the corresponding edge. We formally define the *query motif instance* as well as the *query motif support* as follows:

Definition 2 (Query Motif Instance)**.** *Given the set of query labels L_q, a query motif instance w.r.t L_q is a triangle formed by nodes v_i, v_j and v_k in G, denoted as $\triangle_{ijk}$, such that $v_i, v_j, v_k \in V(L_q)$. The label weight of a query motif instance is defined as*

$$w(\triangle_{ijk}) = \prod_{u \in \{v_i, v_j, v_k\}} |L_q \cap L(u)|$$

Definition 3 (Query Motif Support)**.** *The query motif support of an edge (v_i, v_j) w.r.t L_q in G, denoted as $sup(v_i, v_j, L_q)$, is the sum of label weights of query motif instances containing (v_i, v_j), i.e.,*

$$sup(v_i, v_j, L_q) = \sum_{v_k \in \mathcal{N}(v_i) \cap \mathcal{N}(v_j)} w(\triangle_{ijk})$$

The query motif support of an edge e is the sum of label weights from all triangle motifs containing e. We employ the triangle rather than other more complex motifs because: (1) Triangles in different networks represent stable connections between nodes and are universal building blocks of clusters [35], [36], whereas other motifs are usually only meaningful in certain networks. Thus, we select triangles in this work to avoid finetuning the motifs for different networks. Although higher-order cliques can also represents stable connections, they are very rare in sparse networks and thus cannot achieve good effectiveness in real-world networks which are often sparse. The experiments show that our methods achieve significant results with triangles in different networks. (2) Triangles can be computed efficiently in large graphs [37], [38] whereas it is quite time-consuming to detect complex motifs in large graphs.

Furthermore, the building block of a desired *labeled* cluster should be a triangle of nodes with matching query labels. Hence, the query motif support is an effective method to capture label density and structure density simultaneously.

We can directly use the query motif support to set the weight of e to increase the structure density of the desired labeled cluster H^*. Nonetheless, using triangle motifs alone will result in graph fragmentation for unlabeled graphs [39]. The issue of fragmentation gets worse with our proposed query motif definition, as only a fraction of nodes may contain some query labels. The fragmentation leads to many disconnected components of the underlying graphs, which could even divide H^* into smaller subgraphs and make it more challenging for detecting H^*. Thus, we combine the original graph with the query motif support to define the LAM graph.

Definition 4 (The LAM Graph)**.** *Given a labeled graph G, the set of query labels L_q and a real number $\lambda \in (0, 1)$, the LAM graph G_M is a weighted graph such that $V(G_M) = V(G)$ and $E(G_M) = E(G)$. The weight of an edge $(v_i, v_j) \in E(G_M)$ is defined as:*

$$w_M(v_i, v_j) = \lambda \cdot sup(v_i, v_j, L_q) + (1 - \lambda).$$

In the LAM graph, we denote $d_M = \sum_{v \in \mathcal{N}(u)} w_M(u, v)$ as the weighted degree of u.

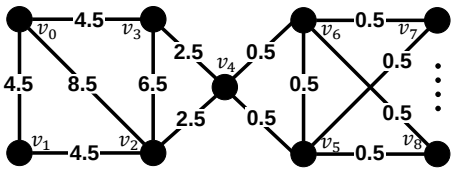


Fig. 2: The LAM graph G_M given query labels $L_q = \{\texttt{IR}, \texttt{DB}\}$ and $\lambda = 0.5$.

The hyperparameter λ is used to balance between the query motifs and the original graph structure. In our experiments, we show that it is easy to find a consistent λ across different datasets for LAM, which validates the user-friendly character of LAM. In contrast, for topology-driven methods, different hyperparameters have to be finetuned for even every query from the same datasets.

Based on the LAM graph, the LAM conductance is defined as the following.

Definition 5 (The LAM Conductance)**.** *The LAM conductance is defined as the weighted conductance in the LAM graph G_M:*

$$\phi_M(H) = \frac{\sum_{(u,v)\in\partial(H)} w_M(u,v)}{\min(\mathrm{vol}_M(H), \mathrm{vol}_M(V) - \mathrm{vol}_M(H))}$$

for a cluster H. The volume $\mathrm{vol}_M(H)$ in G_M is defined as $\mathrm{vol}_M(H) = \sum_{u\in H}\sum_{v\in\mathcal{N}(u)} w_M(u,v)$.

The LAM conductance measures the structure density of H with the consideration of both query labels and triangle motifs. Intuitively, when $\lambda = 1$ and a query with a single label l_q, one can show that $\phi_M(H)$ represents the ratio of the number of query motifs cut across the boundary of H over the total number of query motifs of H. Thus, when $\phi_M(H)$ is small, the nodes within H are more densely connected with query motifs and indicates a desired local cluster w.r.t. the query. We will further conduct an in-depth theoretical analysis to show that LAM can reveal and distinguish the desired cluster in the next subsection.

Example 3. *In Fig. 1, given the query labels $L_q = \{\texttt{IR}, \texttt{DB}\}$ and a triangle $\triangle_{234}$ formed by v_2, v_3 and v_4 as the query motif, its label weight is $w(\triangle_{234}) = 4$. The query motif support of edge (v_2, v_3) equals to 12, since it is contained in two query motifs $\triangle_{234}$ with label weight 4 and $\triangle_{023}$ with label weight 8. Let $\lambda = 0.5$, the weight of (v_2, v_3) in the LAM graph G_M is $w_M(v_2, v_3) = 0.5\times12+(1-0.5) = 6.5$. Fig. 2 shows the LAM graph G_M induced by $L_q = \{\texttt{IR}, \texttt{DB}\}$ and $\lambda = 0.5$. The LAM conductance of the desired cluster $H^* = \{v_0, v_1, v_2, v_3\}$ is $\phi_M(H^*) = \frac{5}{62} = 0.08$. Note that the unweighted conductance of H^* is $\phi(H^*) = \frac{2}{12} = 0.17 > 0.08$. Hence, the desired cluster becomes more densely connected under LAM.*

B. Theoretical Properties of LAM

In this section, we show that the LAM framework can effectively concentrate the personalized pagerank (PPR) distribution from the seed node within the desired cluster, i.e. the PPR values of the nodes within the desired cluster on LAM graph G_M is larger than the corresponding PPR values in the original graph G. Notably, our analysis is also applicable to other random walk ranking models such as the heat-kernel pagerank [16] and the inversed pagerank [17].

The above theoretical guarantee of LAM leads to an easier clustering task. Local graph clustering algorithms [9], [15] commonly utilize the PPR distribution to extract the cluster around the seed node to minimize conductance, and a concentrated PPR distribution can better distinguish the desired cluster from the rest of the graph. For example, [15] extracts the local cluster by identifying sharp drops of the PPR distribution on the boundary of the cluster. By concentrating the PPR distribution, LAM effectively widens the PPR gap between nodes inside and outside the desired cluster, which hence leads to an easier clustering task.

We next formally define the PPR distribution on graphs. Given the seed node s and a decay factor $\alpha \in (0, 1)$, a random walk with restart from s is a traversal of G that starts from s, and at each step, either (i) stops at the current node with α probability, or (ii) goes forward to a randomly selected neighbor of the current node. In this paper, we set $\alpha = 0.1$ by default. For any node $v \in V$, the personalized pagerank (PPR) $\pi(s, v)$ of v w.r.t. s is the probability that a random walk from s terminates at v. Formally, $\pi(s)$ is defined as the linear combination of probabilities that random walks with different lengths terminates at certain nodes and the lengths of random walks are sampled from a geometric distribution: $\pi(s) = \alpha \cdot \mathbf{s}\sum_{t=0}^{\infty}(1-\alpha)^t \cdot P^t$ where P is the transition probability matrix of G and $\mathbf{s}$ is the indicator vector of s. In this section, following previous works [40], [9], [15], we consider the approximate PPR distribution which is defined as

$$\widetilde{\pi}(s) = \alpha\mathbf{s}\sum_{t=0}^{T}(1-\alpha)^t \cdot P^t$$

for a positive integer T as the maximum walk length. Note that the contribution of t-step random walks decays exponentially as t increases. Thus, a good approximation can be achieved with a relatively small T.

For our theoretical analysis, we employ the stochastic block model [40] to generate random graphs of two clusters. Since the stochastic block model does not consider labels, we assign all nodes in one cluster (the desired cluster) with a label l_q. The rest nodes in the graph is assigned with label l_q with a probability. We formalize the random graph model as follows.

Definition 6 (Random Graph)**.** *We denote a random labeled graph $G \sim S(N, 2, p_{in}, p_{out}, p_q)$ as a simple graph with two clusters generated by the stochastic block model. Each cluster contains exactly N nodes. For each pair of nodes from the same cluster, an edge exists with probability p_{in}; for each pair of nodes from different clusters, an edge exists with probability p_{out} and $p_{in} > p_{out}$. Nodes in cluster H_0 have label l_q, and nodes in cluster H_1 are assigned with l_q with probability p_q.*

We denote the set of nodes with label l_q in H_1 as H_{1+}, and the set of nodes without label l_q in H_1 as H_{1-}. We take any node $s \in H_0$ as the seed node and label l_q as the query label. Thus, the desired cluster of the query should be H_0 and

the following theorem shows that LAM concentrates the PPR distribution within H_0 asymptotically.

Theorem 1. *Let $G \sim S(N, 2, p_{in}, p_{out}, p_q)$, and $\pi(H_0)$ and $\pi_M(H_0)$ denote the expectation PPR value of a node in H_0 on G and G_M respectively. For any $\delta > 0$, there exists a sufficiently large N such that the following holds with at least $1 - \delta$ probability:*

$$\widetilde{\pi}_M(H_0) > \widetilde{\pi}(H_0)$$

for the approximated PPR distribution $\widetilde{\pi}$ with any maximum random walk length $T > 0$.

To prove the theorem, note that the PPR distribution is a linear combination of the terminating distribution of random walks with different lengths. Thus, the high-level idea of the proof is to show that for random walks with any fixed length, the probability for such a random walk terminates at a node within H_0 on G_M is larger than the probability on G, as stated in Lemma 2.

The terminating distribution of random walks with any fixed length is decided by the transition probabilities between H_0 and $H_{1\pm}$, which is further decided by the degree distribution between them. Thus, we first give the following lemma that for each node bounds the degree of connection to H_i in $G \sim S(N, 2, p_{in}, p_{out}, p_q)$ and the corresponding G_M.

Lemma 1. *Let $d^j(u)$ and $d^j_M(u)$ be the degree of u connected to H_j in G and G_M respectively. For any $\gamma, \delta > 0$ there exists a sufficiently large N such that*

$$d^j(u) \in [(1-\gamma)\bar{d}^j(u), (1+\gamma)\bar{d}^j(u)], \forall u, \forall j \in \{0, 1+, 1-\}$$

$$d^j_M(u) \in [(1-\gamma)\bar{d}^j_M(u), (1+\gamma)\bar{d}^j_M(u)], \forall u, \forall j \in \{0, 1+, 1-\}$$

holds simultaneously with a probability of at least $1 - \delta$, where node u has in expectation $\bar{d}^j(u)$ and $\bar{d}^j_M(u)$ degree of connection to H_j on G and G_M respectively.

Proof. The proof on graph G can be achieved using Chernoff bounds [41] and union bounds directly since each edge in G exists independently to each other. However, the proof for G_M is more intricate. When query labeled triangles (i.e., triangles with nodes having l_q) are used to set the edge weights, the edge weights in G_M could become correlated.

To this end, we first give concentration bounds on the number of query labeled wedges in G between any pair of nodes u and v, denoted as Λ_{uv}. Note that for two different nodes a and b, they will form a wedge between u and v independently since there is no edge overlap between such two wedges. Thus, by Chernoff bounds we have for any $\gamma_1 > 0$ and any pair of nodes u and v:

$$\Pr\Big(\Lambda_{uv} \notin [(1-\gamma_1)\bar{\Lambda}_{uv}, (1+\gamma_1)\bar{\Lambda}_{uv}]\Big) \leq O(e^{-N})$$

Then, by union bounds, we have:

$$\Pr\Big(\exists u, v : \Lambda_{uv} \notin [(1-\gamma_1)\bar{\Lambda}_{uv}, (1+\gamma_1)\bar{\Lambda}_{uv}]\Big) \leq O(N^2 e^{-N})$$

On the other hand, by Chernoff bounds and union bounds, for any $\gamma_2 > 0$, we also have:

$$\Pr\Big(\exists u : d^j(u) \notin [(1-\gamma_2)\bar{d}^j(u), (1+\gamma_2)\bar{d}^j(u)]\Big) \leq O(Ne^{-N})$$

According to the definition of G_M we have:

$$d^j_M(u) = \sum_{v \in H_j} \Big[(1-\lambda)A_{uv} + \lambda \cdot \Lambda_{uv} \cdot A_{uv}\Big]$$

where A is the adjacent matrix of G.

By the above three formulas and union bounds, we can get:

$$d^j_M(u) \geq (1-\lambda) \cdot (1-\gamma_2)\bar{d}^j(u) + \lambda \cdot (1-\gamma_2)\bar{d}^j(u) \cdot (1-\gamma_1)\bar{\Lambda}_u(H_j)$$

with a probability of at least $1 - O(N^2 e^{-N})$, where $\bar{\Lambda}_u(H_j)$ is the expectation of total number of query labeled wedges formed between u and nodes within H_j. Notice that $\bar{d}^j_M(u) = (1-\lambda)\bar{d}^j(u) + \lambda\bar{d}^j(u)\bar{\Lambda}_u(H_j)$, thus we get:

$$d^j_M(u) \geq (1-\gamma)\bar{d}^j_M(u) + (\gamma - \gamma_2)\bar{d}^j_M(u) + \gamma_1(\gamma_2 - 1)\bar{d}^j(u)\bar{\Lambda}_u(H_j)$$

which implies $d^j_M(u) \geq (1-\gamma)\bar{d}^j_M(u)$ given:

$$\gamma \geq \gamma_1(1-\gamma_2)\frac{\bar{d}^j(u)\bar{\Lambda}_u(H_j)}{\bar{d}^j_M(u)} + \gamma_2$$

Since γ_1 and γ_2 can be arbitrarily small, γ can also be arbitrarily small. An upper bound can be derived similarly, which leads to:

$$\Pr\Big(d^j_M(u) \in [(1-\gamma)\bar{d}^j_M(u), (1+\gamma)\bar{d}^j_M(u)]\Big) \geq 1 - O(N^2 e^{-N})$$

for any u. By union bounds we further get:

$$\Pr\Big(d^j_M(u) \in [(1-\gamma)\bar{d}^j_M(u), (1+\gamma)\bar{d}^j_M(u)], \forall u, \forall j\Big) \geq 1 - O(N^3 e^{-N})$$

which implies Lemma 1 on G_M. □

Note that $\bar{d}^j_M(u)$ and $\bar{d}^j(u)$ keep constant as long as u is in the same H_i. Thus, we denote the $\bar{d}^j(u)$ for $u \in H_i$ as $\bar{d}^i(H_j)$ and $\bar{d}^j_M(u)$ for $u \in H_i$ as $\bar{d}^i_M(H_j)$. We further use $\bar{D}(H_j)$ and $\bar{D}_M(H_j)$ to denote the average degree of nodes within H_j on G and G_M.

Based on Lemma 1, we can prove the following lemma which directly lead to Theorem 1.

Lemma 2. *Given any integer T, let $r_t(H_i)(r^M_t(H_i))$ denote the probability that a t-step random walk terminates at a certain node in H_i on $G(G_M)$ seeded at $s \in H_0$. For any $\delta > 0$, there is an N sufficiently large such that with a probability of at least $1 - \delta$:*

$$r^M_t(H_0) > r_t(H_0)$$

for all $0 < t \leq T$.

Proof. Let $R_t(H_i) = r_t(H_i) \cdot |H_i|$ and $R^M_t(H_i) = r^M_t(H_i) \cdot |H_i|$ denote the probabilities for a t-step random walk terminates within H_i on G and G_M seeded at $s \in H_0$. Based on Lemma 1, we can prove the Equations 1 and 2 using a similar approach described in the first section of appendix for [40]. We omit the details here due to the space limitation.

For any $\epsilon, \delta > 0$, any $i \in \{0, 1+, 1-\}$ and any $T \in \mathbb{N}^+$, there is an N sufficiently large such that:

$$R_t(H_i) \in \left[(1-\epsilon)\bar{R}_t(H_i), (1+\epsilon)\bar{R}_t(H_i)\right] \quad (1)$$

$$R_t^M(H_i) \in \left[(1-\epsilon)\bar{R}_t^M(H_i), (1+\epsilon)\bar{R}_t^M(H_i)\right] \quad (2)$$

holds with a probability of at least $1-\delta$ for all $0 < t \leq T$, where $\bar{R}_t(H_i)$ and $\bar{R}_t^M(H_i)$ are the solutions to the recurrence relation:

$$\bar{R}_t(H_i) = \sum_j \frac{\bar{d}^i(H_j)}{\bar{D}(H_j)} \bar{R}_{t-1}(H_j)$$

$$\bar{R}_t^M(H_i) = \sum_j \frac{\bar{d}_M^i(H_j)}{\bar{D}_M(H_j)} \bar{R}_{t-1}^M(H_j)$$

with $\bar{R}_0(H_0) = \bar{R}_0^M(H_0) = 1, \bar{R}_0(H_{1\pm}) = \bar{R}_0^M(H_{1\pm}) = 0$.

With Equations 1 and 2, Lemma 2 is equivalent to: for any $0 < t \leq T$ and $0 < \lambda < 1$, $\bar{R}_t^M(H_0) > \bar{R}_t(H_0)$.

Let $p = \frac{p_{in}}{p_{out}} > 1$. As $\frac{\bar{d}_M^{1-}(H_j)}{\bar{D}_M(H_j)} = O(N^{-1})$ for $j \in \{0, 1+\}$, we can omit them given N is sufficiently large. Thus, $\bar{R}_t^M(H_{1-}) = \left(\frac{\bar{d}_M^{1-}(H_{1-})}{\bar{D}_M(H_{1-})}\right)^t \bar{R}_0^M(H_{1-}) = 0$. By further omitting quantities that are $O(N^{-1})$, we can derive:

$$\begin{aligned}\bar{R}_t^M(H_0) &= \frac{\bar{d}_M^0(H_0)}{\bar{D}_M(H_0)} \bar{R}_{t-1}^M(H_0) + \frac{\bar{d}_M^0(H_{1+})}{\bar{D}_M(H_{1+})} \bar{R}_{t-1}^M(H_{1+}) \\ &= \frac{(p^2+p_q)\bar{R}_{t-1}^M(H_0)}{p^2+2p_q+p_q^2} + \frac{(1+p_q)(1-\bar{R}_{t-1}^M(H_0))}{1+2p_q+p^2p_q^2}\end{aligned}$$

Let $a = \frac{(p^2+p_q)}{p^2+2p_q+p_q^2} - \frac{(1+p_q)}{1+2p_q+p^2p_q^2}$, $b = \frac{(1+p_q)}{1+2p_q+p^2p_q^2}$, we have:

$$\bar{R}_t^M(H_0) = a\bar{R}_{t-1}^M(H_0) + b$$

which implies:

$$\bar{R}_t^M(H_0) = a^t + \frac{b}{1-a}(1-a^t)$$

We can observe that $\bar{R}_t^M(H_0)$ is not influenced by λ when N is sufficiently large. Similarly, we can get

$$\bar{R}_t(H_0) = c^t + \frac{d}{1-c}(1-c^t)$$

given $c = \frac{p-1}{p+1}$, $d = \frac{1}{p+1}$. It's easy to check that $a > c$ and $\frac{d}{1-c} < \frac{b}{1-a} < 1$. Thus, we have:

$$\begin{aligned}\bar{R}_t(H_0) &= c^t + \frac{d}{1-c}(1-c^t) < a^t + \frac{d}{1-c}(1-a^t) \\ &< a^t + \frac{b}{1-a}(1-a^t) = \bar{R}_t^M(H_0)\end{aligned}$$

for which we complete the proof. □

Apart from the above theoretical analysis, as to be introduced later in our experiment, we further evaluate the PPR distribution on real-world datasets with ground-truth clusters and verify that LAM framework concentrates the PPR distribution within the desired ground-truth clusters empirically. As shown in Table II, when the LAM framework is applied, the PPR value within ground-truth clusters consistently becomes larger across all datasets.

TABLE II: Comparison of PPR distribution on the LAM graph G_M and the original graph G. We use $\bar{\pi}_M$ to denote the average amount of PPR value within ground-truth clusters on G_M and $\bar{\pi}$ to denote the corresponding value on G. Note that the seed node for each ground-truth cluster is selected randomly within the ground-truth cluster.

Dataset	$\bar{\pi}$	$\bar{\pi}_M$	Relative Increase
facebook	0.51	0.6052	18.7%
amazon	0.865	0.9886	14.3%
dblp	0.5944	0.8841	48.7%
youtube	0.31	0.431	39%
livejournal	0.6115	0.9538	55.98%
orkut	0.2773	0.8755	215.7%

IV. Index-free Local Clustering

In this section, we first introduce the cluster metric based on the LAM framework. Subsequently, we devise the index-free algorithm for local clustering over labeled graphs by optimizing the proposed cluster metric. Note that our algorithm only requires local access of the graph around the seed node. The trick is to only construct the necessary edges in G_M on the fly instead of constructing the whole G_M.

A. Cluster Metric based on LAM

As presented in Sec. II-A, the cluster metric $f_q(H)$ simultaneously considers the query label density and structure density of cluster H. We introduce a general cluster metric based on the LAM conductance $\phi_M(H)$:

$$f_q(H) = \frac{\rho(H, L_q)}{\phi_M(H)}$$

The above metric $f_q(H)$ uses $\phi_M(H)$ to measure the structure density as LAM effectively increases structure density within the desired cluster as shown in Sec. III-B. The function $\rho(H, L_q)$ is used to measures the label density of H and a suitable definition of $\rho(H, L_q)$ is crucial for the effectiveness of $f_q(H)$. Different datasets and queries require different $\rho(H, L_q)$ to achieve best effectiveness. In this paper, we consider two kinds of label score functions. The SOTA labeled community search algorithm ATC [7] adopt the label score function $\rho_1(H, L_q) = \sum_{l \in L_q} \frac{|V(l) \cap H|^2}{|H|}$. We also use another label score function $\rho_2(H, L_q) = \sum_{l \in L_q} \frac{|V(l) \cap H|}{|H|}$, which measures the average number of matched query labels in H.

Note that it is NP-hard to optimize $f_q(H)$. The proof is trivial when considering a subset of instances of the problem. For unlabeled graphs, we have $\rho(H, L_q) = 0$ constantly for any H and L_q. Thus, the problem is equivalent to conductance minimization problem, which is a well-known NP-hard problem [42].

B. Index-free Two-stage Algorithm

Given that maximizing $f_q(H)$ is NP-hard, in this section, we devise a two-stage heuristic algorithm to maximize $f_q(H)$. Note that the objective function $f_q(H)$ is a fraction with $\rho(H, L_q)$ as its numerator and $\phi_M(H)$ as its denominator. In stage I, the algorithm obtains a candidate cluster as an initial solution by minimizing the LAM conductance $\phi_M(H)$. Then,

Algorithm 1: LAM Clustering

Input: Graph G, the seed node s and the set of query labels L_q, error tolerance ϵ, size bound b

Output: The subgraph H^* with maximum $f_q(\cdot)$

1 $\pi_M(s,u) \leftarrow 0$ for each u ; // Stage I
2 $r(s,u) \leftarrow 0$ for each $u \neq s$ and $r(s,s) \leftarrow 1$;
3 **while** $\exists u : r(s,u) > \epsilon \cdot d(u)$ **do**
4 $\quad \pi_M(s,u) \leftarrow \pi_M(s,u) + \alpha \cdot r(s,u)$;
5 $\quad r(s,u) \leftarrow (1-\alpha) \cdot \frac{r(s,u)}{2}$;
6 $\quad$ for each $v \in \mathcal{N}(u)$:
$\quad\quad r(s,v) \leftarrow r(s,v) + (1-\alpha) \cdot \frac{r(s,u) \cdot w_M(u,v)}{2 \cdot d_M(u)}$;
7 **end**
8 for each u with $\pi_M(s,u) > 0$: $\pi_M(s,u) \leftarrow \frac{\pi_M(s,u)}{d(u)}$;
9 sort nodes by descending $\pi_M(s)$ and u_i denotes the i-th node after sorting;
10 $H \leftarrow \arg\min_{H_i \forall i \leq b} \phi_M(H_i)$, where $H_i = \{u_1, ..., u_i\}$;
11 $\rho \leftarrow 0$; // Stage II
12 **while** $\rho < \rho(H, L_q)$ **do**
13 $\quad \rho \leftarrow \rho(H, L_q)$;
14 $\quad v_r \leftarrow \arg\min_{u \in H} \mathrm{dep}(u, H)$;
15 $\quad$ remove v_r from H;
16 **end**
17 $H^* \leftarrow H \cup \{v_r\}$, where v_r is the last removed node;
18 **return** H^*

in stage II, it iteratively trims the nodes from the initial solution to maximize $\rho(H, L_q)$.

In the two-stage algorithm, $\phi_M(H)$ is minimized first, after which $\rho(H, L_q)$ is maximized through a peeling process. There are two major benefits to choose this optimization order. First, minimizing $\phi_M(H)$ first can obtain a good initial solution as $\phi_M(H)$ has already incorporated the label information in the processing. In contrast, maximizing $\rho(H, L_q)$ first may lead to find a cluster with nodes having query labels but poorly connected, which has undesirably low structure density. Second, there exist many theoretically guaranteed algorithms to obtain clusters with low un/weighted conductance scores [9], [15], [43]. Furthermore, these algorithms are index-free and efficient to be executed on large graphs. Hence, we can simply apply these existing algorithms to obtain a good initial solution without reinventing the wheels. The pseudo code of the two-stage algorithm is presented in Alg. 1. It takes graph G, a seed node s, a set of query labels L_q and error tolerance ϵ for computing the personalized pagerank (PPR) as inputs, and outputs a cluster H^*. We next describe the details of the algorithm as follows.

Stage I: minimizing $\phi_M(H)$ (Lines 1-10). We adopt the sweep algorithm [9] for minimizing $\phi_M(H)$. The sweep algorithm first requires calculating the approximate PPR distribution around the seed node over G_M. To avoid constructing the whole G_M, we extend the LocalPush (Line 1-7) through lazy construction of G_M, i.e. it only constructs the adjacent edges of nodes when they are going to be pushed. According to [9], the LocalPush algorithm computes $\pi_M(s)$ by accessing at most $O(\frac{1}{\epsilon})$ edges. Hence, we can efficiently obtain the PPR distribution over G_M with the LocalPush extension and only a local construction of G_M.

Subsequently, the nodes are sorted by the normalized PPR values in a descending order (Line 8-9). Finally, a "sweep" operation is employed to examine first b *prefixes* in the sorted node list. It computes the LAM conductance $\phi_M(H)$ of each prefix and returns the one with the smallest $\phi_M(H)$ as the output cluster of stage I (Line 10). The size bound b is used to avoid the free-rider effect as mentioned in [44] and we set $b = 1000$ in this work.

Stage II: maximizing $\rho(H, L_q)$ (Lines 11-17). We further refine the result by peeling unpromising nodes to optimize for $\rho(H, L_q)$. Note that we should consider the impact of node removals on $\phi_M(H)$, which determines the overall score $f_q(H)$. However, finding the set of nodes R that maximizes $f_q(H \setminus R)$ is again NP-hard. To this end, we adopt a greedy framework that removes nodes iteratively from the initial solution H to form smaller candidate clusters with higher label density. In each iteration, the algorithm computes a score, namely *density perturbation* $\mathrm{dep}(u, H)$, for each node u in H. This density perturbation $\mathrm{dep}(u, H)$ measures the impact on $\phi_M(H)$ and $\rho(H, L_q)$ of removing node u from H simultaneously. Then, the node v_r, which has the minimum score $\mathrm{dep}(v_r, H)$, is iteratively removed from H until the $\rho(H, L_q)$ cannot be increased anymore.

It is crucial to define a proper $\mathrm{dep}(u, H)$. For a node $u \in H$, $\mathrm{dep}(u, H)$ should be large if removing u from H leads to substantially higher LAM conductance. Meanwhile, $\mathrm{dep}(u, H)$ should be large if removing u from H reduces $\rho(H, L_q)$ significantly. Therefore, we define $\mathrm{dep}(u, H)$ as:

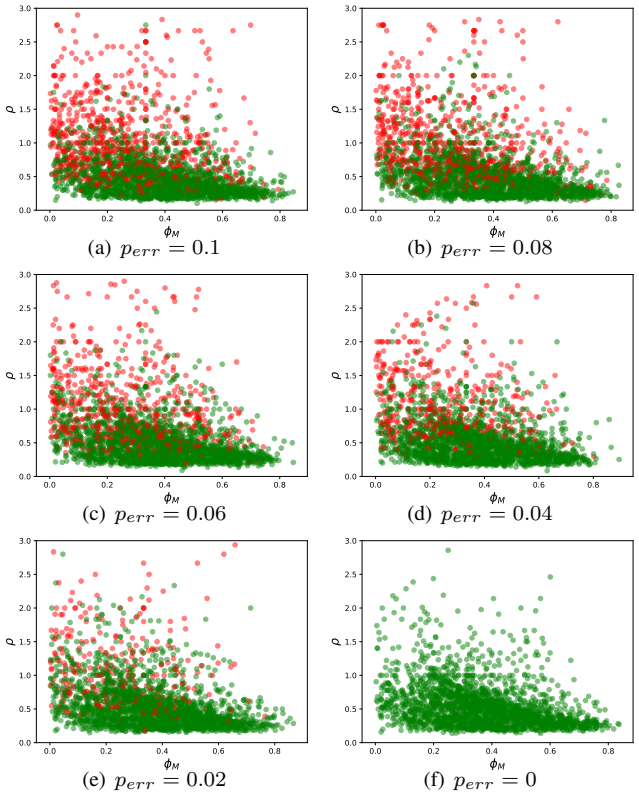
$$\mathrm{dep}(u, H) = \Delta\rho \cdot \frac{\phi_M(H - \{u\})}{\phi_M(H)}$$

The first factor of $\mathrm{dep}(u, H)$ measures the impact of removing u to the label density $\rho(H, L_q)$. For example, $\Delta\rho = |L(u) \cap L_q|$ for ρ_2. The second factor is the ratio between the LAM conductance before and after removing u.

Now it comes to computing the $\phi_M(H - \{u\})$ for all $u \in H$ in each iteration. Note that $\phi_M(H - \{u\})$ is equal to:

$$\frac{d_{in}(u, H) - d_{out}(u, H) + \sum_{(u,v) \in \partial(H)} w_M(u, v)}{\mathrm{vol}_M(H) - d_M(u)}$$

where $d_{in}(u, H)$ is the weighted degree of u connected to nodes within H, i.e. $d_{in}(u, H) = \sum_{v \in \mathcal{N}(u) \cap H} w_M(u, v)$; $d_{out}(u, H) = d_M(u) - d_{in}(u, H)$ is the weighted degree of u connected to nodes outside H. Computing $\mathrm{dep}(u, H)$ directly requires traversing the neighbors of u and counting $d_{in}(u, H)$ and $d_{out}(u, H)$ respectively, which takes $O(d(u))$ time per iteration. We use an $O(|H|)$ additional space and a simple pre-processing in each iteration to compute $\mathrm{dep}(u, H)$ efficiently. At the beginning of stage II, we initialize $d_{in}(u, H)$ and $d_{out}(u, H)$ for each node $u \in H$, which takes $O(|H|)$ space. Then, at each iteration, removing node v_r only affects



(a) $p_{err} = 0.1$ (b) $p_{err} = 0.08$ (c) $p_{err} = 0.06$ (d) $p_{err} = 0.04$ (e) $p_{err} = 0.02$ (f) $p_{err} = 0$

Fig. 3: Preferred label score function for queries on `youtube` database with different label noise (Refer to Sec. V-A). The horizontal axis represents the label-aware motif conductance (ϕ_M) and the vertical axis represents the average number of matched query labels(ρ). Each point represents a query. A red point indicates that $\rho_1(H, L_q)$ is preferred by the query; a green point indicates that $\rho_2(H, L_q)$ is preferred.

$d_{in}(u, H)$ and $d_{out}(u, H)$ of $u \in N(v_r) \cap H$, which is updated by traversing the neighbors of v_r only. Thereby, $\text{dep}(u, H - \{v_r\})$ is computed in $O(1)$ by loading the $d_{in}(u, H - \{v_r\})$ and $d_{out}(u, H - \{v_r\})$ values.

Complexity Analysis. Let $d_{max} = \max_{u \in V} d(u)$ denote the maximum node degree in G, and $L_{max} = \max_u |L(u)|$ represent the maximum number of labels attached to a node. The following theorem gives the time complexity of Alg. 1.

Theorem 2. *Alg. 1 terminates in $O(\frac{d_{max} \cdot L_{max}}{\epsilon})$ time.*

Proof. We first give the time complexity of obtaining the approximate PPR distribution over G_M. The key point here is to calculate the complexity for computing $sup(v_i, v_j, L_q)$ of an edge (v_i, v_j) that will be accessed during LocalPush. The support $sup(v_i, v_j, L_q)$ is obtained by counting all the query motifs containing (v_i, v_j) and their weights, which requires computing the common neighbors of v_i and v_j. We assume that the neighbor list of each node is sorted, $\mathcal{N}(v_i) \cap \mathcal{N}(v_j)$ can be computed in $O(d(v_1) + d(v_2)) = O(d_{max})$ time. To obtain the query motif weights, for each $u \in \mathcal{N}(v_i) \cap \mathcal{N}(v_j)$, the algorithm computes $|L(u) \cap L_q|$ by scanning $L(u)$, which is bounded by $O(L_{max})$. In summary, the time complexity for computing $sup(v_i, v_j, L_q)$ is $O(d_{max} \cdot L_{max})$. Furthermore, the LocalPush algorithm will access $O(\frac{1}{\epsilon})$ edges. Thus, the time complexity for computing PPR distribution over G_M is $O(d_{max} \cdot L_{max}) \cdot O(\frac{1}{\epsilon}) = O(\frac{d_{max} \cdot L_{max}}{\epsilon})$.

The algorithm then sorts nodes according to normalized PPR value. Note that we only obtain the ordered list of the b nodes with the largest normalized PPR value among nodes that are pushed (i.e. with PPR value larger than 0) in the algorithm. The number of nodes that are pushed is also bounded by $O(\frac{1}{\epsilon})$. Thus, this operation can be implemented with a time complexity of $O(\frac{1}{\epsilon} \cdot \log b)$ through a max heap. The last step of Stage I requires computing the conductance of each *prefix* in the sorted node list. The time complexity of this step is bounded by the volume of the first b nodes, which is $O(d_{max} \cdot b)$. Since the size bound b is set to be a constant number, the total time complexity of Stage I is bounded by $O(\frac{d_{max} \cdot L_{max}}{\epsilon}) + O(\frac{1}{\epsilon} \cdot \log b) + O(d_{max} \cdot b) = O(\frac{d_{max} \cdot L_{max}}{\epsilon})$.

We then dicuss the time complexity of Stage II. Denoting the output cluster of Stage I as H, the initialization of $d_{in}(u, H)$ and $d_{out}(u, H)$ for each node u requires a traversal of neighbors of u, and thus takes $O(\text{vol}(H))$ time. In any iteration, the computation of $\text{dep}(u, H)$ for each node u takes $O(|H|)$ time. Given that there are at most $|H|$ iterations, the time complexity of removing irrelevant nodes until the algorithm terminates is $O(|H|^2)$. Thus, the total time taken by stage II is bounded by $O(|H|^2 + \text{vol}(H))$. Obviously, we have $|H| \leq b$ and $\text{vol}(H) \leq d_{max} \cdot b$, which means the time complexity of Stage II is constantly bounded.

Thus, the time complexity of Alg. 1 is bounded by the time complexity of Stage I, which is $O(\frac{d_{max} \cdot L_{max}}{\epsilon})$. □

According to Theorem 2, by setting $\epsilon = \Omega(\frac{1}{n})$ where $n = |V|$ is the number of nodes in G, we can get a sublinear algorithm for local clustering on labeled graphs. To further demonstrate the efficiency of our proposed framework, in the experiments, we set $\epsilon = \Theta(\frac{1}{n})$ and the experimental results show that even under this setting, our proposed algorithm can achieve the local clustering on labeled graphs efficiently.

C. Suitable Label Score Function Selection

The label score function $\rho(H, L_q)$ is used in Stage II in our two-stage algorithm for refining the candidate cluster generated in Stage I. Hence, a suitable label score function is crucial for the effectiveness of our algorithm. Different datasets and queries will prefer different label score functions. Thus, in this subsection, we further propose a linear classifier to decide for a query whether $\rho_1(H, L_q)$ or $\rho_2(H, L_q)$ should be used.

Through experiments, we can find that the label-aware motif conductance and the average number of matched query labels of the coarse candidate cluster generated in stage I are effective features for the classifier. Fig. 3 presents the relationship between the performance of two label score functions, i.e. $\rho_1(H, L_q)$ and $\rho_2(H, L_q)$, and the two features in `youtube` with different label distributions as an example. We can

observe that queries that prefer $\rho_1(H, L_q)$ (red points) and $\rho_2(H, L_q)$ (green points) can be divided quite well through a linear classifier.

Therefore, we choose the linear classifier as the model for deciding the proper label score function for each query. Specifically, to train this classifier, we randomly sample 100 queries on each dataset. We use Q to denote the set of queries that are sampled for training. For each $q \in Q$, we use $y_1(q)$ to denote the performance of Alg. 1 for q with $\rho_1(H, L_q)$ and $y_2(q)$ to denote the performance of Alg. 1 with $\rho_2(H, L_q)$. We use Q^+ to denote the set of queries such that $y_2(q) > y_1(q)$ and use Q^- to denote the set of queries such that $y_2(q) < y_1(q)$.

We train the linear classifier $\mathcal{C}$ by minimizing the following straightforward loss function based on the LSE (least square error).

$$Loss = \sum_{q \in Q^+ \wedge \mathcal{C}(q)=1} (y_2(q) - y_1(q))^2 - \sum_{q \in Q^- \wedge \mathcal{C}(q)=1} (y_2(q) - y_1(q))^2 - \sum_{q \in Q^+ \wedge \mathcal{C}(q)=2} (y_2(q) - y_1(q))^2 + \sum_{q \in Q^- \wedge \mathcal{C}(q)=2} (y_2(q) - y_1(q))^2 \quad (3)$$

In the loss function, $\mathcal{C}(q) = 1$ denotes that the classifier outputs $\rho_1(H, L_q)$ for the query q; $\mathcal{C}(q) = 2$ denotes that the classifier outputs $\rho_2(H, L_q)$ for the query q.

For query processing, we first execute the stage I of Alg. 1 and obtain the values of the two required features by the linear classifier. We then use the classifier to choose the suitable label score function to be used in the peeling process of stage II.

V. Experiments

We evaluate the LAM framework with the two-stage algorithm through experiments on both real-world and sythetic datasets. Sec. V-A describes the setup. Subsequent subsections are presented to answer the following research questions:

- Can LAM outperform the SOTA solutions for identifying desired clusters in both real and synthetic datasets (Sec. V-B)?
- Is LAM robust to different label distributions (Section V-C)?
- Does the peeling process in the two-stage algorithm indeed boost the results (Sec. V-D)?
- Is the index-free algorithm efficient and scalable enough to handle large graphs (Sec. V-E)?
- Can queries with different labels reveal diverse yet densely-connected clusters (Sec. V-F)?
- Is parameter tuning made easy for the LAM (Section V-G)?

A. Experimental Setup

Datasets. Eight real-world datasets are used in our experiments: `facebook`, `amazon`, `dblp`, `youtube`, `livejournal`, `orkut`, `wikidata` and `friendster`. We use the first 6 datasets for effectiveness study, and use the last 2 datasets for efficiency and scalability study. The dataset statistics are reported in Table III.

The `facebook` dataset contains 10 ego-networks for ego-users $X \in \{0, 107, 348, 414, 686, 698, 1684, 1912, 3437, 3980\}$ with both ground-truth clusters and real-world labels. For a given user X, the ego-network of X is the induced subgraph of the facebook network by X and its neighbors. Each node in the `facebook` dataset has labels representing features like political stand and education degree. The label values are anonymized to numbers for privacy concern. In our experiments, we aggregate the results of 10 ego-networks within `facebook` for ease of presentation when doing efficiency study and parameter tuning.

TABLE III: Dataset Statistics. Φ gives the average unweighted conductance of ground-truth clusters in the network. `sbm` is generated by the stochastic block model with $\mu = 0.9$.

Network	$\|V\|$	$\|E\|$	d_{max}	Φ	Labeled
`fb0`	347	2519	77	0.59	✔
`fb107`	1045	26749	253	0.59	✔
`fb348`	227	3192	99	0.52	✔
`fb414`	755	1693	57	0.45	✔
`fb686`	170	1656	77	0.52	✔
`fb698`	66	270	29	0.265	✔
`fb1684`	792	14024	136	0.29	✔
`fb1912`	755	30025	293	0.76	✔
`fb3437`	547	4813	107	0.87	✔
`fb3980`	59	146	18	0.415	✔
`amazon`	335K	926K	549	0.07	✘
`dblp`	317K	1M	342	0.414	✘
`youtube`	1.1M	3M	28754	0.84	✘
`livejournal`	4M	35M	14815	0.395	✘
`orkut`	3.1M	117M	33313	0.73	✘
`friendster`	65M	1.8B	5214	0.76	✘
`wikidata`	20.6M	75M	2.9M	-	✔
`sbm`	500	125K	288	0.9	✘

The `amazon`, `dblp`, `youtube`, `livejournal`, `orkut` and `friendster` are real-world networks contain ground-truth clusters but do not contain real-world labels. To this end, we follow the setup used in [7] to augment labels for these datasets. We generate a label set consisting of $|\mathcal{L}| = \eta \cdot |V|$ distinct labels for each network with $\eta = 0.0001$ by default. For each ground-truth cluster, we randomly select 3 labels from $\mathcal{L}$ as the representative labels for this cluster and assign each of these labels to nodes in the cluster with probability $(1 - p_{err})$. We set $p_{err} = 0$ by default. Furthermore, to model noise in the label data, for each node v in the network, we randomly assign 1 to 5 labels to v uniformly.

The `wikidata` is a large knowledge graph (KG) extracted from *wikidata*[2]. Each node represents an entity in the KG and has different types of relationships with other entities. The node types are node categories from the KG ontology.

We also include synthetic random graphs generated by stochastic block model $S(N, k, p_{in}, p_{out})$, where the mixing ratio $\mu = \frac{(k-1)p_{out}}{p_{in}+(k-1)p_{out}}$ is used to measure the fraction of neighbors of each node that belong to different clusters. In our experiments, we set $k = 10$, $N = 50$, $p_{in} = 0.5$, and vary μ from 0.1 to 0.9 to generate different random graphs. The labels are generated for these synthetic graphs as for real-world networks with $\eta = 0.02$ by default.

Query Generation. We generate one query for each ground-truth cluster contained in each network. The seed node is

[2]https://dumps.wikimedia.org/wikidatawiki/entities/

TABLE IV: Parameters used in experiments.

Param.	Description	Default
μ	The mixing ratio of the stochastic block model.	0.9
η	The ratio between the total number of generated labels to $\|V\|$.	1e-4, 2e-2
p_{err}	The probability of label noise within ground-truth clusters.	0.0
λ	The hyperparameter in LAM.	0.4

randomly selected from the cluster to avoid the influence of different seeding algorithms [7]. Note that seed selection is orthogonal to our work. The query labels are selected for each query as the representative labels for the corresponding cluster. The representative labels for unlabeled real-world networks and synthetic random networks are the labels assigned to the cluster, described in the label generation process. For `facebook` ego-networks with real-world labels, we use top-3 labels that occurring most frequently in a given cluster and rarely occuring outside the cluster as representative labels [7].

Compared Methods. We compared the proposed methods with the SOTA methods on label-aware community search.

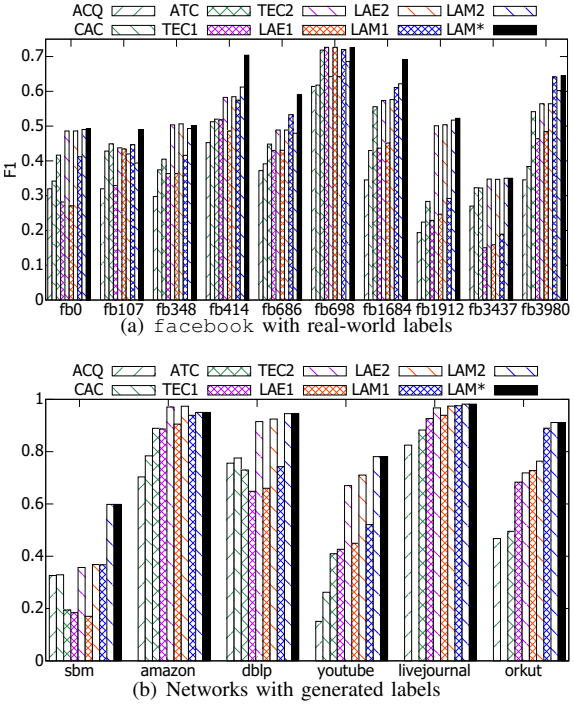
- ATC [7] returns a (k, d)-truss community containing the seed node and maximizes the label density $\rho_1(H, L_q)$.
- ACQ [6] finds a k-core community containing the seed node that maximizes the number of labels shared in L_q by all nodes in the returned community.
- CAC [23] finds a triangle-connected k-truss containing the seed node that maximizes the number of labels shared in L_q by all nodes in the returned community.

Note that for ACQ, CAC and ATC, suitable hyperparameters k and d have to be assigned for each query. We follow the setup in [7] and tune k for each query to be the maximum value that can return a non-empty community for ATC and ACQ. The value of d for ATC is set as the same in [7].

Variants of Our Approach. We compare different variants of our approach for the ablation study.

- TEC1 uses unweighted conductance with $\rho(H, L_q) = \rho_1(H, L_q)$ and employs the two-stage algorithm to extract the cluster.
- TEC2 is a variant of TEC1 with $\rho(H, L_q) = \rho_2(H, L_q)$.
- LAE1 uses edge weighted conductance with $\rho(H, L_q) = \rho_1(H, L_q)$ and employs the two-stage algorithm to extract the cluster. An edge weight is assigned as 1 if it has two side nodes with query labels, and $(1 - \lambda)$ otherwise.
- LAE2 is a variant of LAE1 with $\rho(H, L_q) = \rho_2(H, L_q)$.
- LAM1 is our proposed approach with the LAM conductance and $\rho(H, L_q) = \rho_1(H, L_q)$ used in $f_q(H)$.
- LAM2 is a variant of LAM1 with $\rho(H, L_q) = \rho_2(H, L_q)$.
- LAM* uses a linear classifier to determine whether $\rho_1(H, L_q)$ or $\rho_2(H, L_q)$ shoudl be used as the label score function, as described in Sec. IV-C.

Parameter Setup. The parameters used in the experiments and their default values are summarized in Table IV. We set the hyperparameter $\lambda = 0.4$ during our experiments, under which LAM based methods can consistently achieve the best overall result across all datasets (Sec. V-G).



(a) `facebook` with real-world labels

(b) Networks with generated labels

Fig. 4: Effectiveness comparison for queries with single seed.

Evaluation Metric. We use the F1-score, which is the harmonic average of precision and recall w.r.t. the ground-truth clusters to validate the *effectiveness* of different algorithms. For evaluating the *efficiency*, we consider the average of time and memory consumption for answering the queries.

Environment. All experiments are conducted on a Linux Server with Intel Xeon Gold 6140 CPU and 256 GB memory running Ubuntu 18.04. We use the original implementation of ATC [7], CAC [23] (implemented with C++) and ACQ [6] (implemented with Java) provided by the authors. Other algorithms are implemented with C++ and executed with a single thread.

B. Effectiveness Evaluation

We evaluate the overall effectiveness of our proposed methods based on the LAM framework (LAM1, LAM2 and LAM*) against both the topology-driven methods (ATC, ACQ and CAC) and variants of our approaches for the ablation study (TEC1, TEC2, LAE1 and LAE2). Fig. 4 reports the F1 scores of the compared methods on each dataset for queries with single seed node and LAM* **achieves the best effectiveness over most datasets.** Note that the results of CAC on `livejournal` and `orkut` are omitted since CAC cannot finish index construction within 100 hours. We further make the following two observations from the results.

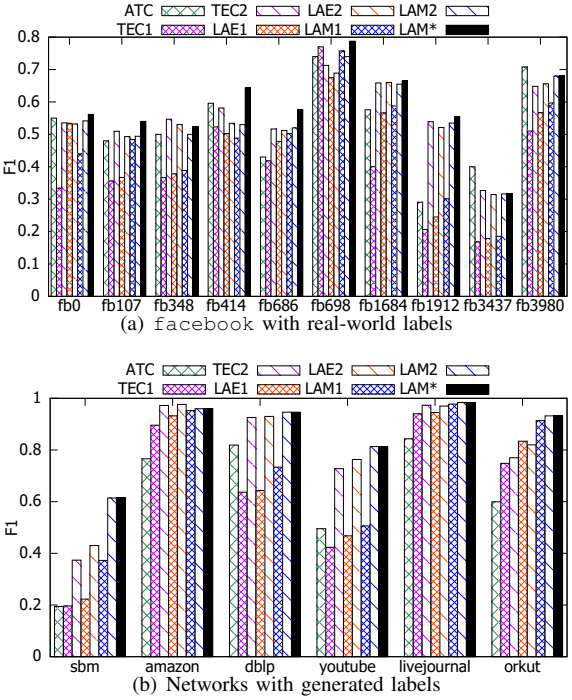
First, on networks with generated labels (Fig. 4(b)), LAM* achieves the same effectiveness as LAM2; on ego-networks with real-world labels from `facebook`, LAM* can achieve better effectiveness than both LAM1 and LAM2. This is due to different distributions of generated labels and real-world

labels. The generated labels are well distributed within the ground-truth clusters for each query, and thus all queries from these networks prefer $\rho_2(H, L_q)$ over $\rho_1(H, L_q)$; in contrast, real-world labels on ego-networks from facebook (Fig. 4(a)) contain much more noises, LAM* can achieve better performance than LAM1 and LAM2 by automatically choosing a suitable label score function for each query.

Second, both LAM1 and LAM2 outperform the corresponding TEC1, TEC2 and LAE1, LAE2 in most cases, which shows that our LAM framework can indeed enhance the effectiveness of the proposed two-stage algorithm regardless of different choices of $\rho(H, L_q)$. Specifically, LAM1 outperforms TEC1 and LAE1 on 9 out of 10 facebook networks, LAM2 outperforms TEC2 and LAE2 on 7 out of 10 facebook networks. On networks with generated labels, only the effectiveness of LAM2 is slightly worse than TEC2 and LAE2 on amazon. This is still due to the well-structured ground-truth clusters of amazon, which makes it easy to extract the clusters solely based on unweighted conductance or label-aware edge weighted conductance.

We also conduct experiments for queries with multiple seed nodes. Our method only requires one seed node as input. Thus, for queries with multiple seed nodes, we lauch a query for each seed node respectively and then return the community that maximize the score function $f_q(H)$. Fig. 5 and Fig. 6 present the effectiveness results for queries with 2 and 3 seed nodes respectively. Note that ACQ and CAC cannot support queries with multiple seed nodes, thus the results for ACQ and CAC are omitted. We can observe that LAM* still perform better than ATC and other variants of the two-stage algorithm for queries with multiple seed nodes over most datasets. Note that the result of ATC for queries with 3 seed nodes on orkut is omitted from Fig. 6(b) since it takes more than 3 days for ATC to finish 1000 queries. Nevertheless, in the following experiments, we only consider queries with single seed node by default since it's usually easier for users to specify just one node to kick start the clustering task in practice. Besides, the querying time for ATC increases significantly with the increase of number of seed nodes, whereas the querying time for our method only increases slightly when multiple seed nodes are selected.

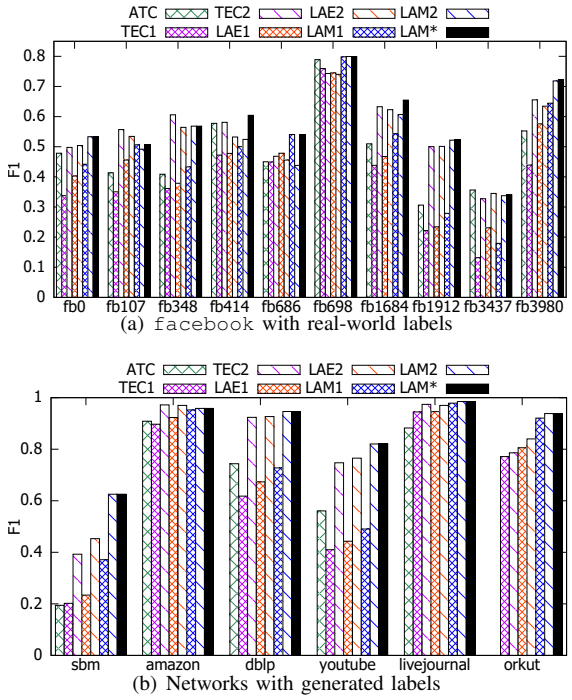
We further use the synthetic networks with different values of mixing ratio μ on the sbm dataset to evaluate all compared methods. A larger μ implies that the number of edges between clusters increases which thereby makes the clusters harder to extract. The experimental result is shown in Fig. 7. Overall, LAM* consistently outperform the baselines with various values of μ. When μ is varied from 0.1 to 0.6, LAM* can almost recover the ground clusters exactly; when μ is varied from 0.6 to 0.8, the effectiveness of LAM* decreases slightly; when μ is further increased to 0.9, the F1 score of LAM* incurs a relatively sharp drop, but still achieves significantly better performance than the baselines. The F1 score of all other baselines, except for TEC2 and LAE2, incur a sharp drop with a smaller value of μ; TEC2 and LAE2 incur a sharper drop than LAM* when μ is increased to 0.9. This observation indicates that LAM* is more robust to structural noises than the baselines.



(a) facebook with real-world labels

(b) Networks with generated labels

Fig. 5: Effectiveness comparison for queries with 2 seeds.



(a) facebook with real-world labels

(b) Networks with generated labels

Fig. 6: Effectiveness comparison for queries with 3 seeds.

C. Label Distribution Robustness

In this section, we vary the parameters p_{err} and η of label assignments to evaluate the robustness of all methods to different label distributions.

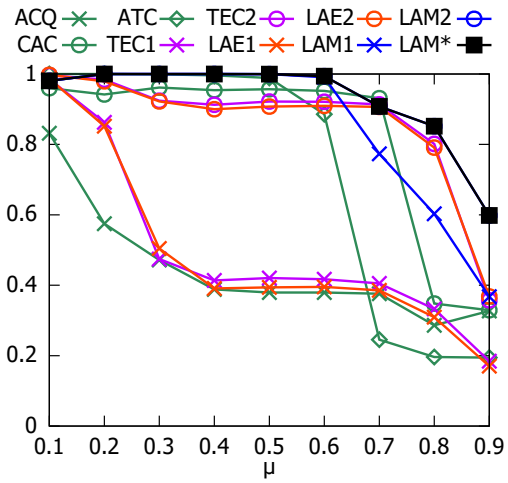


Fig. 7: Effectiveness on synthetic networks.

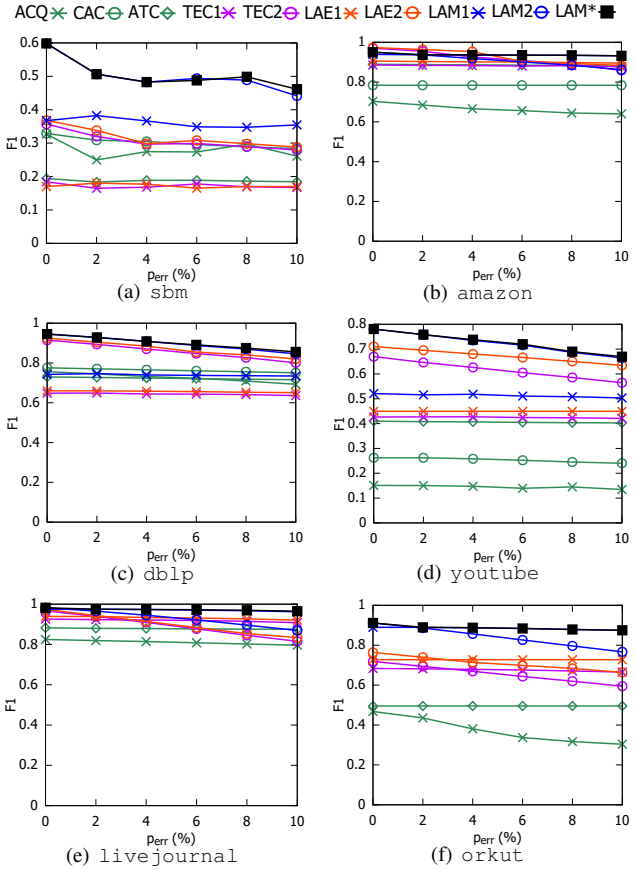


Fig. 8: Varying p_{err}.

Varying p_{err}. The parameter p_{err} measures the query label distribution within each ground-truth cluster, i.e., the probability that a node within the ground-truth cluster is assigned with the desired labels of the queries. We vary p_{err} from 2% to 10%. Figure 8 reports the F1 scores of all compared methods when p_{err} is varied. The scores drop in general as there are less query labels within the ground-truth cluster. Our proposed method LAM* still achieves consistently better results than those produced by the topology-driven community search methods as well as other variants of the two-stage algorithm over all datasets.

We also see that LAM2 is relatively more sensitive to query label distributions within the ground-truth clusters than LAM1. This is due to different label score functions $\rho(H, L_q)$ that are used in LAM1 and LAM2. The label score function $\rho_1(H, L_q) = \sum_{l \in L_q} \frac{|V(l) \cap H|^2}{|H|}$, which is quadratic to the count of query labels within H, tends to extract H with a large count of query labels but possibly small query label density. Thus, when the query label within H is relatively infrequent, it can still be extracted by LAM1 since H is still the cluster that contains the largest count of query labels around the seed. On the other hand, $\rho_2(H, L_q) = \sum_{l \in L_q} \frac{|V(l) \cap H|}{|H|}$ is linear to the count of query labels within H, which means LAM2 will extensively trim the nodes that are not assigned with query labels so that the query label density is maximized. Nevertheless, LAM* is more robust than both LAM1 and LAM2 since it can select the proper label score function for each query.

Varying η. The parameter η measures the query label distribution outside each ground-truth cluster, i.e., the probability that a node outside the ground-truth cluster H is assigned with the same labels of H. We vary η from 1e-4 to 5e-4 for real-world datasets, and vary η from 0.02 to 0.1 for the synthetic network generated by stochastic block model. Figure 9 reports the F1 scores with varying η. As in the previous experiments, our LAM* consistently outperform other baselines.

We also observe that LAM1 is more sensitive to query label distributions outside the ground-truth cluster (LAM1 requires a larger η to achieve good performance whereas LAM2 performs well with small η). This is consistent with the properties of $\rho(H, L_q)$ used in LAM1 and LAM2. The label score function $\rho_1(H, L_q)$ tends to include nodes outside the ground-truth cluster that are assigned with query labels, whereas LAM2 will trim these nodes from the result cluster.

D. Ablation Study of the Peeling Process

In this section, we conduct an ablation study on the effectiveness of the peeling process in Stage II of our algorithm. We use LAM$^-$, LAE$^-$ and TEC$^-$ to represent the algorithm that only optimizes the corresponding conductance through Stage I without conducting the peeling process in Stage II. We report the results on datasets where ground-truth communities are present.

Table V(a) reports the F1-score of LAM$^-$ compared with corresponding algorithms LAM1 and LAM2. The results show that both the peeling process in Stage II based on $\rho_1(H, L_q)$ and $\rho_2(H, L_q)$ can boost the performance of local clustering significantly on labeled graphs. Similar observations can be obtained form Table V(b) and V(c) for LAE$^-$ and TEC$^-$.

E. Efficiency and Scalability

In this section, we compare the efficiency and scalability between topology-driven methods (ACQ, CAC and ATC) and

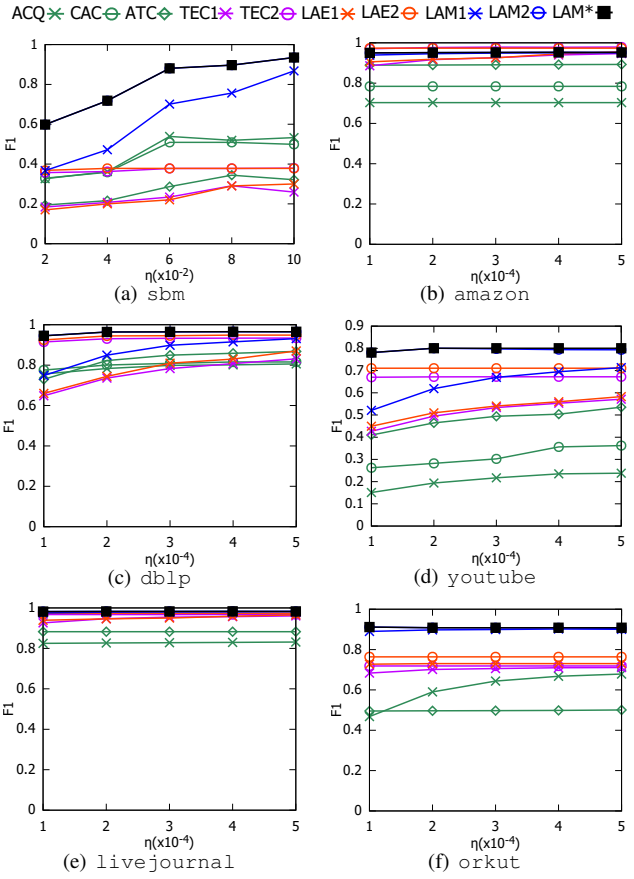


Fig. 9: Varying η.

(a) LAM

Dataset	LAM$^-$	LAM1	LAM2
facebook	0.48	0.484	0.525
amazon	0.90	0.939	0.95
dblp	0.61	0.743	0.945
youtube	0.26	0.52	0.78
livejournal	0.75	0.975	0.98
orkut	0.79	0.89	0.91
sbm	0.36	0.37	0.598

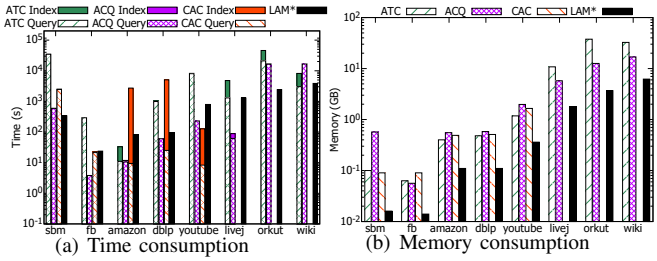
(b) LAE

Dataset	LAE$^-$	LAE1	LAE2
facebook	0.386	0.405	0.512
amazon	0.8	0.905	0.97
dblp	0.53	0.66	0.925
youtube	0.24	0.45	0.71
livejournal	0.75	0.94	0.97
orkut	0.57	0.73	0.76
sbm	0.16	0.17	0.37

(c) TEC

Dataset	TEC$^-$	TEC1	TEC2
facebook	0.39	0.393	0.51
amazon	0.53	0.886	0.97
dblp	0.42	0.648	0.914
youtube	0.21	0.427	0.67
livejournal	0.63	0.93	0.967
orkut	0.45	0.68	0.72
sbm	0.16	0.184	0.36

TABLE V: Ablation Study of the Peeling Process.



(a) Time consumption (b) Memory consumption

Fig. 10: Time and memory consumption.

our conductance-driven approaches (LAM*). The purpose is to demonstrate the efficacy of our index-free algorithm.

Time Consumption. Fig. 10(a) compares the running time of LAM* and topology-driven methods. Note that all topology-driven methods need to construct indexes before processing queries. Thus, we compare the running time of 1000 queries for LAM* with the running time of 1000 queries plus time of index construction for topology-driven methods.

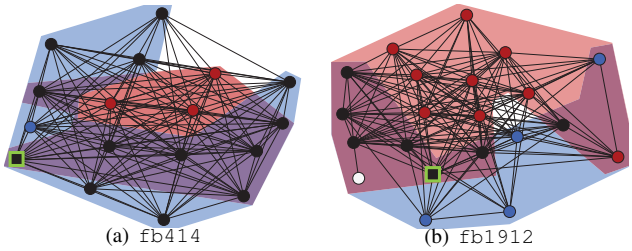
We note that ATC is still less efficient than LAM* even with the help of indexes over most datasets. The queries of CAC is faster than LAM* with the help of indexes. However, the index construction of CAC is extremely time-consuming. Note that CAC even cannot finish index construction in reasonable time on livejournal, orkut and wikidata. ACQ is more efficient than LAM* on relatively small datasets but is still more time consuming than LAM* when the networks grow larger (orkut and wikidata). Nevertheless, the effectiveness of ACQ is inferior among the compared methods.

Memory Consumption. Fig. 10(b) reports the memory consumption of LAM* and topology-driven methods. Due to large indexes required by topology-driven baselines. Topology-driven baselines consume significantly more memory than index-free LAM*. The gap on memory consumption between LAM* and topology-driven baselines scales w.r.t. the size of the network. For example, on small networks facebook, dblp, youtube, ATC requires 3 to 4 times memory than LAM*. On larger networks livejournal, orkut, ATC requires up to 10 times memory than LAM*. This implies that our methods are much more efficient than topology-driven baselines in terms of memory consumption.

Scalability. We further compare the scalability of LAM* with topology-driven baselines using the extremely large network friendster. All topology-driven methods cannot handle queries from friendster since they fail to construct the indexes with 256GB memory on our machine. In contrast, our method can handle queries from friendster with 60GB memory. We randomly select 100 queries from friendster and it takes on average 50 seconds to process a query for LAM*. Furthermore, the performance can be easily accelerated by parallel processing [45]. Thereby, our methods based on LAM is scalable to handle large networks.

F. A Case Study on facebook

In this subsection, we further validate that LAM* can extract diverse (different clusters will be detected for different query



(a) fb414 (b) fb1912

Fig. 11: Examples of clusters returned with different query labels. In each ego-network of facebook, two queries are issued from the highlighted seed in the green square with query labels l_1 and l_2 respectively. The subgraph shaded with red (blue) is the cluster returned for label l_1 (l_2). The color of each node represents its labels. Red nodes are associated with l_1; blue nodes are associated with l_2; black nodes have both labels; the white node has neither labels.

labels) yet densely-connected clusters through a case study on facebook dataset.

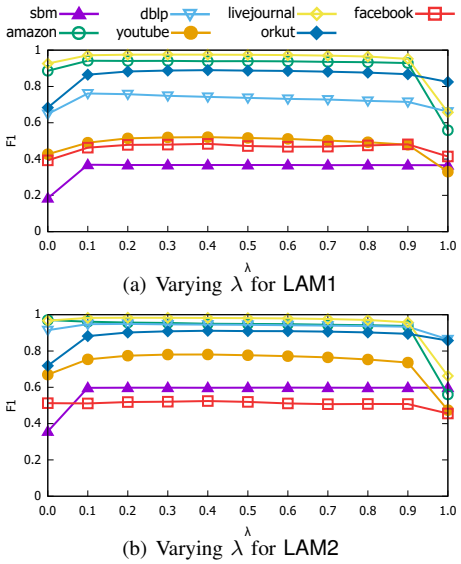
In Fig. 11, we visualize two clusters extracted by different query labels l_1 and l_2 from ego-networks fb414 and fb1912. The query labels l_1 and l_2 are top-2 representative labels for the corresponding ground-truth clusters. In fb414, we can see that both clusters extracted for query label l_1 and l_2 have high structure density. For the query label l_2, the returned cluster (with blue shade) includes nodes that contain label l_2 (note that black nodes are attached with both labels), while nodes with only label l_1 is excluded. In contrast, using l_1 as the query label extracts a cluster containing nodes with label l_1 (nodes colored in red). Note that several nodes with l_1 are also excluded by LAM* for the red-shaded cluster to achieve higher structure density. In fb1912, with l_1 as the query label, the red-shaded cluster is returned. Note that a blue node with only label l_2 is included to achieve higher structure density. When the query label is l_2 , the blue-shaded cluster that are highly related to label l_2 is extracted. The observation shows that using local clustering queries with different labels can reveal label-aware cohesive clusters.

G. Parameter Tuning for λ in LAM

We vary the hyperparameter λ used in the LAM framework and report the F1 scores of LAM1 and LAM2 in Figure 12.

We have the following observations. First, the performance is relatively stable when λ varies from 0.1 to 0.9 across all datasets for both LAM1 and LAM2, and drops dramatically when λ equals to 0 or 1. The reason is that when $\lambda = 0$, the LAM framework is equivalent to the unweighted conductance and does not consider the motif and label information at all; whereas when $\lambda = 1$, LAM suffers from the fragmentation issue. Note that fragmentation is not observed on the sbm dataset when $\lambda = 1$. This is because the clusters in sbm are well connected with each other due to the large μ, and thus do not suffer from fragmentation.

Second, with the increase of λ, the F1 scores of both LAM1 and LAM2 slightly decrease on amazon, dblp and livejournal, while first increase and then decrease on youtube and orkut. With a larger λ, the LAM framework



(a) Varying λ for LAM1

(b) Varying λ for LAM2

Fig. 12: Parameter Tuning.

places more emphasis on the query motif, which results in the PPR distribution to be more concentrated around the seed node. Hence, on amazon, dblp and livejournal with well-structured ground-truth clusters (i.e., small average unweighted conductance as shown in Table III), the PPR distribution is very concentrated even with a relatively small λ. In this case, LAM would omit some relevant nodes and cause the slight drop on the F1 scores. On the other hand, youtube and orkut have illy-structured clusters (i.e., large average unweighted conductance). Focusing on the query motif can help LAM avoid nodes outside the cluster, and thus the F1 scores increase when λ is varied from 0.1 to 0.4. However, if λ continues to increase, the PPR distribution becomes too concentrated and causes slight performance drops. The fluctuation of the F1 scores on ego-networks from facebook is less regular than that on other datasets. This is because the distribution of real-world labels is not as ideal as that of labels generated synthetically for other datasets. Nevertheless, the F1 scores of both LAM1 and LAM2 on facebook is relatively stable when λ is varied.

According to the observations in Figure 12, we can see that a consistent $\lambda = 0.4$ for all queries across all datasets achieves the best result overall. The results have confirmed the design of the LAM framework, which strikes a balance between the original graph and the adjusted edge weights by the query motif instances.

VI. Conclusion

In this paper, we studied the problem of local clustering over labeled graphs. We observed that some existing topology-driven labeled community search approaches can be adapted to our problem. However, these methods suffer from the requirement of strict topology constraints and prohibitively large indexes. We thus proposed a novel LAM framework and devised the index-free two-stage algorithm for local clustering

on labeled graphs. Theoretically, we have proved that this framework can concentrate the PPR values within the desired cluster, which is a desired property for local graph clustering. Compared with prior community search methods, our methods can find clusters with more complex structures by extending the search space of the candidate cluster, and thus achieves better effectiveness across all datasets in the experiments.

REFERENCES

[1] M. Girvan and M. E. Newman, "Community structure in social and biological networks," *Proceedings of the national academy of sciences*, vol. 99, no. 12, pp. 7821–7826, 2002.

[2] E. M. Daly and M. Haahr, "Social network analysis for routing in disconnected delay-tolerant manets," in *MOBIHOC*, 2007, pp. 32–40.

[3] X. Li, M. Wu, C.-K. Kwoh, and S.-K. Ng, "Computational approaches for detecting protein complexes from protein interaction networks: a survey," *BMC genomics*, vol. 11, no. 1, pp. 1–19, 2010.

[4] G. Palla, I. Derényi, I. Farkas, and T. Vicsek, "Uncovering the overlapping community structure of complex networks in nature and society," *nature*, vol. 435, no. 7043, pp. 814–818, 2005.

[5] E. L. Huttlin, R. J. Bruckner, J. A. Paulo, J. R. Cannon, L. Ting, K. Baltier, G. Colby, F. Gebreab, M. P. Gygi, H. Parzen *et al.*, "Architecture of the human interactome defines protein communities and disease networks," *Nature*, vol. 545, no. 7655, pp. 505–509, 2017.

[6] Y. Fang, R. Cheng, S. Luo, and J. Hu, "Effective community search for large attributed graphs," *Proceedings of the VLDB Endowment*, vol. 9, no. 12, pp. 1233–1244, 2016.

[7] X. Huang and L. V. Lakshmanan, "Attribute-driven community search," *Proceedings of the VLDB Endowment*, vol. 10, no. 9, pp. 949–960, 2017.

[8] T. Van Laarhoven and E. Marchiori, "Local network community detection with continuous optimization of conductance and weighted kernel k-means," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 5148–5175, 2016.

[9] R. Andersen, F. Chung, and K. Lang, "Local graph partitioning using pagerank vectors," in *FOCS*, 2006, pp. 475–486.

[10] D. F. Gleich and C. Seshadhri, "Vertex neighborhoods, low conductance cuts, and good seeds for local community methods," in *SIGKDD*, 2012, pp. 597–605.

[11] H. Yin, A. R. Benson, J. Leskovec, and D. F. Gleich, "Local higher-order graph clustering," in *SIGKDD*, 2017, pp. 555–564.

[12] C. E. Tsourakakis, J. Pachocki, and M. Mitzenmacher, "Scalable motif-aware graph clustering," in *WWW*, 2017, pp. 1451–1460.

[13] A. Arenas, A. Fernandez, S. Fortunato, and S. Gomez, "Motif-based communities in complex networks," *Journal of Physics A: Mathematical and Theoretical*, vol. 41, no. 22, p. 224001, 2008.

[14] L. Huang, C.-D. Wang, and H.-Y. Chao, "A harmonic motif modularity approach for multi-layer network community detection," in *ICDM*, 2018, pp. 1043–1048.

[15] R. Andersen and F. Chung, "Detecting sharp drops in pagerank and a simplified local partitioning algorithm," in *TAMC*, 2007, pp. 1–12.

[16] K. Kloster and D. F. Gleich, "Heat kernel based community detection," in *SIGKDD*, 2014, pp. 1386–1395.

[17] P. Li, I. Chien, and O. Milenkovic, "Optimizing generalized pagerank methods for seed-expansion community detection," in *NeurIPS*, 2019, pp. 11 710–11 721.

[18] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 22, no. 8, pp. 888–905, 2000.

[19] Y. Li, K. He, K. Kloster, D. Bindel, and J. Hopcroft, "Local spectral clustering for overlapping community detection," *ACM transactions on knowledge discovery from data*, vol. 12, no. 2, pp. 1–27, 2018.

[20] H. Yin, A. R. Benson, and J. Leskovec, "The local closure coefficient: A new perspective on network clustering," in *WSDM*, 2019, pp. 303–311.

[21] F. Moradi, T. Olovsson, and P. Tsigas, "A local seed selection algorithm for overlapping community detection," in *ASONAM*, 2014, pp. 1–8.

[22] X. Huang, L. V. Lakshmanan, and J. Xu, "Community search over big graphs: Models, algorithms, and opportunities," in *ICDE*, 2017, pp. 1451–1454.

[23] Y. Zhu, J. He, J. Ye, L. Qin, X. Huang, and J. X. Yu, "When structure meets keywords: Cohesive attributed community search," in *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, 2020, pp. 1913–1922.

[24] Y. Zhu, Q. Zhang, L. Qin, L. Chang, and J. X. Yu, "Querying cohesive subgraphs by keywords," in *ICDE*, 2018, pp. 1324–1327.

[25] L. Chen, C. Liu, K. Liao, J. Li, and R. Zhou, "Contextual community search over large social networks," in *ICDE*, 2019, pp. 88–99.

[26] Z. Zhang, X. Huang, J. Xu, B. Choi, and Z. Shang, "Keyword-centric community search," in *ICDE*, 2019, pp. 422–433.

[27] Q. Liu, Y. Zhu, M. Zhao, X. Huang, J. Xu, and Y. Gao, "Vac: Vertex-centric attributed community search," in *ICDE*, 2020, pp. 937–948.

[28] J. Gao, J. Chen, Z. Li, and J. Zhang, "Ics-gnn: lightweight interactive community search via graph neural network," *Proceedings of the VLDB Endowment*, vol. 14, no. 6, pp. 1006–1018, 2021.

[29] Y. Fang, Y. Yang, W. Zhang, X. Lin, and X. Cao, "Effective and efficient community search over large heterogeneous information networks," *Proceedings of the VLDB Endowment*, vol. 13, no. 6, pp. 854–867, 2020.

[30] X. Huang, H. Cheng, and J. X. Yu, "Dense community detection in multi-valued attributed networks," *Information Sciences*, vol. 314, pp. 77–99, 2015.

[31] Y. Zhou, H. Cheng, and J. X. Yu, "Graph clustering based on structural/attribute similarities," *Proceedings of the VLDB Endowment*, vol. 2, no. 1, pp. 718–729, 2009.

[32] Y. Ruan, D. Fuhry, and S. Parthasarathy, "Efficient community detection in large networks using content and links," in *WWW*, 2013, pp. 1089–1098.

[33] J. Yang, J. McAuley, and J. Leskovec, "Community detection in networks with node attributes," in *ICDM*, 2013, pp. 1151–1156.

[34] C. Zhe, A. Sun, and X. Xiao, "Community detection on large complex attribute network," in *SIGKDD*, 2019, pp. 2041–2049.

[35] C. Seshadhri, T. G. Kolda, and A. Pinar, "Community structure and scale-free collections of erdős-rényi graphs," *Physical Review E*, vol. 85, no. 5, p. 056109, 2012.

[36] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world'networks," *Nature*, vol. 393, no. 6684, pp. 440–442, 1998.

[37] J. Shun and K. Tangwongsan, "Multicore triangle computations without tuning," in *ICDE*, 2015, pp. 149–160.

[38] R. Pearce, "Triangle counting for scale-free graphs at scale in distributed memory," in *HPEC*, 2017, pp. 1–4.

[39] P.-Z. Li, L. Huang, C.-D. Wang, and J.-H. Lai, "Edmot: An edge enhancement approach for motif-aware community detection," in *SIGKDD*, 2019, pp. 479–487.

[40] I. M. Kloumann, J. Ugander, and J. Kleinberg, "Block models and personalized pagerank," *Proceedings of the national academy of sciences*, vol. 114, no. 1, pp. 33–38, 2017.

[41] R. Motwani and P. Raghavan, *Randomized algorithms*. Cambridge university press, 1995.

[42] J. Šíma and S. E. Schaeffer, "On the np-completeness of some graph cluster measures," in *SOFSEM*, 2006, pp. 530–537.

[43] F. Chung, "The heat kernel as the pagerank of a graph," *Proceedings of the National Academy of Sciences*, vol. 104, no. 50, pp. 19 735–19 740, 2007.

[44] Y. Wu, R. Jin, J. Li, and X. Zhang, "Robust local community detection: on free rider effect and its elimination," *Proceedings of the VLDB Endowment*, vol. 8, no. 7, pp. 798–809, 2015.

[45] B. Bahmani, K. Chakrabarti, and D. Xin, "Fast personalized pagerank on mapreduce," in *SIGMOD*, 2011, pp. 973–984.