Index-free Query Processing for Local Clustering on Labeled Graphs: A Motif-aware Approach

Niu Yudong
Singapore Management University
ydniu.2018@phdcs.smu.edu.sg

Yuchen Li
Singapore Management University
yuchenli@smu.edu.sg

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

ABSTRACT

Local clustering is an essential tool to explore large graphs. In this paper, we study the problem of local clustering on labeled graphs, which extracts a subgraph with nodes having high label density matched to query labels as well as high structure density around a seed node. Existing related works suffer from two major limitations: (I) The candidate subgraphs have to comply with strict topology-driven models and better candidates can be pruned by the strict constraints; (II) The topological constraints give rise to substantial computational overheads and existing works have to construct prohibitively large indices for online processing. To address the limitations, we study the conductance-driven local clustering that ensures structure density through minimizing conductance. Conductance is a well-understood metric for detecting unlabeled clusters. However, for labeled graphs, applying conductance directly is insufficient since 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 the weights capture the label and structure proximity of two nodes simultaneously. 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 the nice properties of LAM, we propose an index-free peeling algorithm to efficiently search local clusters on labeled graphs. Extensive experiments on both synthetic and real-world networks demonstrate that our proposed method achieves significantly better effectiveness and scalability than the state-of-the-art methods.

1 INTRODUCTION

The graph model has emerged as a prevalent tool to enable analysis over growing volume and complexity of big data. By modeling relationships between entities, such as persons, genes and transactions, graphs allow efficient data exploration to extract insights for numerous domains [13, 17]. To support group analysis, graph clustering, also known as graph partitioning or community detection, has attracted continuous attention and a plethora of algorithms have been developed [16]. Unlike global graph clustering algorithms, local clustering algorithms are efficient to discover local clusters around a seed node and are scalable for large graph analysis as the local approaches avoid the prohibitive cost of accessing the entire graph [25, 61]. To analyze more clusters of a large graph, one can iteratively find local clusters around a seed node and change the seed node to discover other clusters [55].

Although the local clustering problem has been extensively studied on unlabeled graphs, detecting local clusters on *labeled graphs* has just gained attention recently. Many real-world graphs are *labeled graphs* which associate the nodes with labels. For instance, online social media models users as nodes, user profiles as node labels and friendships/interactions as edges. 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. Another typical example is gene interaction networks, where 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. As labeled graphs can contain a large number of labels and not every label is relevant to a particular user, it thus calls for an efficient and effective approach to find clusters that are relevant to the user's interests.

In this paper, we study the problem of local clustering on labeled graphs: 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.

Prior works. A line of study known as community search on labeled graphs has similar objectives as the local clustering problem studied in this work. However, most existing algorithms for community search on labeled graphs adopt the *topology-driven* models to define a community. These topology-driven models include k-core [1] [14], (k, d)-truss [2] [23] and many others [7, 9, 47, 50, 56]. In particular, a subgraph must satisfy the topological constraints before it can be considered as a candidate community. Albeit effective for some application scenarios, there are two major drawbacks of the topology-driven models:

- *Usability.* Users need to fine-tune the hyperparameter for the community search methods to be effective. For instance, users need to specify a just-nice k value for the k-core model when handling different graphs or even different seed nodes of the same graph [14]. This is because the model only selects subgraphs satisfying the k-core constraints as candidates. Thus, the topological constraints lack flexibility and could easily miss the desired community. Additionally, the topology-driven models pose steep learning curves to users as the models are not intuitive for non-experts.
- *Index Overhead.* It requires substantial computational overheads to identify candidate communities satisfying the topological constraints. Hence, prior approaches have to construct sizable indices for online community search. Unfortunately, both *time* and *space* overheads of the index-based approaches are prohibitive for large graphs. For example, it takes over 7 hours to

[1] Given a graph G and an integer k, a k-core is a subgraph H such that for $\forall v \in H, \deg_H(v) \geq k$, where $\deg_H(v)$ is the degree of v in H.
[2] Given a graph G, integers k and d, a (k, d)-truss is a subgraph H such that for each edge $e \in H, sup_H(e) \geq k$ and the diameter of H is not larger than d, where $sup_H(e)$ is the number of triangles in H contains e.

construct a state-of-the-art index [23] for the `orkut` dataset in our experiments, and the memory consumption of the index is more than 10 times of the graph size. Even worse, the index has to be reconstructed when the underlying graphs are subject to dynamic updates.

Present work. In this paper, we propose to search local clusters on labeled graphs with a conductance-driven approach. Conductance [3, 18, 49], 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. 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 candidate clusters with small conductance values (i.e., high structure density). Nevertheless, a naïve adoption of conductance leads to poor quality on labeled graphs as conductance does not consider the synergy between label and structure information for detecting labeled clusters.

To this end, we propose a novel and intuitive *Label-Aware Motif weighted* framework (LAM). LAM is primarily motivated by prior works for motif-aware graph clustering on unlabeled graphs [4, 20, 48, 55]. Motifs are high-order structures that recur frequently in the underlying graph. Instead of operating directly on the original graph, motif-aware approaches assign a weight to each edge based on how many motif instances containing the edge. Then, the edge weighted graphs are fed to the weighted conductance model for detecting high-order clusters. We generalize this idea to labeled graphs and propose the LAM framework for detecting labeled clusters. Given a set of query labels, we define the concept of *query motif*, as any *triangle* instance where the labels of each node in the triangle have non-zero overlap with the query labels. The query motif is a fundamental unit of labeled clusters with high label as well as structure densities. Subsequently, we set the weight of each edge in the underlying graph according to the number of query motifs containing the edge. In this way, the weight of an edge captures the label and structure proximity between any two nodes simultaneously.

We theoretically show that, with the LAM framework, the personalized pagerank (PPR) distribution will be concentrated within the labeled cluster around the seed node on random graphs generated by the stochastic block model. The concentrated PPR distribution will benefit clustering algorithms as nodes in the desired cluster are boosted with higher PPR values so that they are easily distinguished from nodes outside the cluster.

Built upon the LAM framework, we propose an efficient *index-free* algorithm for local clustering on labeled graphs. As conductance optimization is generally NP-hard [42], we propose a heuristic two-stage algorithm for efficient cluster detection. In the first stage, we employ the sweep algorithm [3] to efficiently extract a cluster as a coarse-grained candidate H_0. Some outliers may be included in this coarse-grained candidate. Hence, we further execute a fine-grained 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, and 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 paper 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 (Section 3). To the best of our knowledge, this is the first work for local clustering driven by the concept of query motifs on labeled graphs.
- We devise a two-stage algorithm based on the LAM framework (Section 4). One highlight of our algorithm is *index-free*, which make processing graphs with billions of edges possible. 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 indices due to the prohibitive memory consumption. In contrast, our method support `friendster` without resorting to costly indices.
- We conduct extensive experiments on both real and synthetic networks to validate the effectiveness as well as efficiency of the proposed approach over the state-of-the-art methods (Section 5). LAM consistently outperforms the baselines and achieves up to 38% absolute improvement on F1 scores. Furthermore, the two-stage algorithm shows competitive efficiency while using 10x fewer memory than the indexed-based approaches.

2 PRELIMINARIES

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

2.1 Notations and Problem Formulation

Notations. In this work, we consider an undirected *labeled* graph $G = (V, E, L)$ with node set V, edge set E and label set 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 $L(u) \subseteq L$. 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 any label $l \in L$, we use $V(l)$ to denote the set of nodes associated with label l, i.e. $V(l) = \{u | L(u) \cap \{l\} \neq \emptyset\}$. In addition, without loss of generality, we consider that graph G is connected.

Given a subset of nodes $H \subseteq V$, $\mathrm{vol}(H) = \sum_{u \in H} d(u)$ denotes the volume of H, and $\partial(H)$ denotes the *edge boundary set* of H, i.e., $\partial(H) = \{(v_i, v_j) \in E \mid v_i \in H, v_j \notin H\}$. Then, the conductance of H in G is defined as:

$$\phi(H) = \frac{|\partial(H)|}{\min(vol(H), vol(G) - vol(H))}.$$

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. For any node $v \in V$, the personalized pagerank (PPR) $\pi(s, v)$ of v wrt. 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

Table 1: Frequently used notations.

Notation	Description
$G = (V, E, L)$	A labeled graph with node set V, edge set E and label set L
$G_M = (V, E, w_M)$	The query-motif aware graph for G
$\mathcal{N}(u)$	The set of neighbors of u
$d(u), d_M(u)$	The degree of u in G and the weighted degree of u in G_M respectively
s	The seed node
L_q	The set of query labels
$L(u)$	The set of labels of u
$\text{vol}(H), \text{vol}_M(H)$	The volume of H in G and G_M
$\phi(H), \phi_M(H)$	The conductance of H on G and G_M
λ	The hyperparameter used in constructing G_M
$V(l)$	The set of nodes that have label l
$G(H)$	The subgraph induced by H. We use H and $G(H)$ interchangeably when the context is clear.

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 the seed node s. In this paper, we consider the approximate PPR distribution like previous works [2, 3, 27], 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 relatively small T.

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 expensive for large graphs [32]. In this paper, we study local clustering on labeled graphs where the clusters are extracted around a seed node. Note that how to select the seed node is orthogonal to this work. Interested users can refer to the existing works on how to pick a good seed node [18, 36, 54].

Definition 1. *Given a labeled graph $G = (V, E, L)$ and a set of query labels $L_q \subseteq L$, 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.

There have been some existing works to propose different versions of metric $f_q(H)$, which will be reviewed in the remaining part of this section.

2.2 Related Work

In this section, we review closely related studies in three categories: (1) local graph clustering; (2) community search; and (3) global graph clustering on labeled graphs.

Local Graph Clustering is an extensively studied area for graph analytics. Various metrics for unlabeled graphs have been proposed. The classic edge density [40] and edge-surplus [47] maximize the internal denseness of the cluster. The subgraph modularity [35], density-isolation [28], unweighted conductance [3, 26, 46] optimize the internal density and external sparseness simultaneously. The local modularity metric [10] aims at sharpening the boundary of the cluster. [52] shows that all these objective functions can suffer from the free-rider effect, and propose the seed node biased density to overcome the issue. Recently, the motif-aware conductance [55] is proposed to take the high-order structures of the network into account. Note that the above methods are designed for unlabeled graphs and ignore the label information.

Community Search aims at finding the community according to the query given by the user, where the communities are defined based on specific topology-driven models. For unlabeled graphs, the query contains a set of query nodes and the community search methods returns a community containing the set of query nodes. Various topology-driven models for undirected graphs have been proposed and studied: core-based models [6, 12, 45], truss-based models [1, 21, 24], clique based models [11, 47, 50, 56] and edge connectivity component (ECC)-based models [7, 19]. For directed graphs, Fang et al.[15] and Liu et al.[33] propose the D-core and D-truss model respectively.

For labeled graphs, several community metrics are proposed recently so that the structure density and label density are considered simultaneously. According to different query inputs, the community search methods on labeled graphs can be further divided into three categories.

The first category receives a set of query labels as inputs [8, 57, 60]. 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. The second category takes a set of query nodes as inputs [34]. The target is to find a community that contains the query nodes while have similar labels within the community. The third category receives both a set of query labels and a set of query nodes as inputs [14, 23]. The target is to find a community that contains the query nodes and has high query label density within the community. The third category of labeled community search methods can be adopted to solve label-aware local clustering problem studied in this work. However, these methods are based on strict topological constraints to define their community models, like core-based models [14] and truss-based models [23]. As discussed in Section 1, the topological constraints can discard better candidates from the solution just because the constraints are not satisfied, and require expensive indices for online processing. In contrast, our proposed method generalizes conductance to find local clusters without imposing hard constraints as well as index construction.

Global Graph Clustering on Labeled Graphs, also known as labeled community detection (LCD), divides a labeled graph into a number of partitions so that each partition is densely connected and the nodes in each partition share similar labels [22, 39, 53, 58, 59]. 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.

3 THE LAM FRAMEWORK

As discussed in the previous section, the existing works design different metrics to find clusters and propose dedicated algorithms to optimize for their designed metrics. In this section, we introduce a Label-Aware Motif weighted framework (LAM) 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 the framework concentrates the PPR distribution of the desired clusters for any seed node in that cluster.

3.1 Transformation under LAM

Motivated by the motif-aware graph clustering on unlabeled graphs [4, 20, 48, 55], 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*

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

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 motif because triangles are fundamental building blocks of clusters [41, 51] and can be computed efficiently in large graphs [38, 43]. 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 only to set the edge weights will result in graph fragmentation for unlabeled graphs [31]. 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^*. To overcome fragmentation, we combine the original graph with the query motif support to define the LAM graph.

Definition 4 (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 the weight of an edge is defined as:*

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

where $\mathbb{1}(v_i, v_j)$ is an indicator if $(v_i, v_j) \in E$. We denote $d_M = \sum_{v \in \mathcal{N}(u)} w_M(u, v)$ as the weighted degree of u.

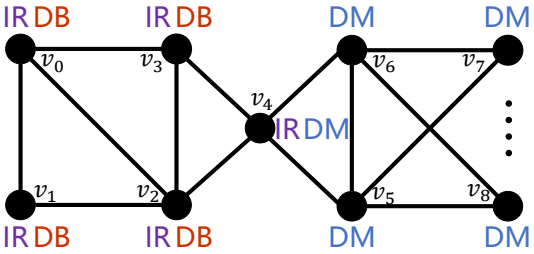

Figure 1: A labeled graph G with the desired cluster $H^* = \{v_0, v_1, v_2, v_3\}$ given the seed node v_0 and query labels {DB, IR}.

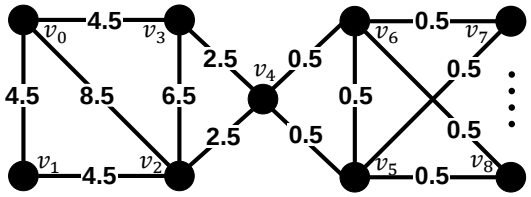

Figure 2: The LAM graph G_M given query labels $L_q = \{\texttt{IR}, \texttt{DB}\}$ and $\lambda = 0.5$.

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.

Based on the LAM graph, we define the LAM conductance.

Definition 5. *The LAM conductance is defined as the weighted conductance:*

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

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

The LAM conductance measures the structure density of H with 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 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 wrt. the query. We will further conduct an in-depth theoretical analysis to show LAM can reveal and distinguish the desired cluster in the next subsection.

Example 1. *Figure 1 displays a small part of an academic collaboration graph G (the dots between v_7 and v_8 represents the remaining nodes in the graph). Each node in G represents a researcher and the labels associated with the node, e.g.,* DB *and* IR*, represents the research areas. Each edge represents a collaboration relationship between two researchers. Given query labels $L_q = \{\texttt{IR}, \texttt{DB}\}$, triangle $\triangle_{234}$ formed by v_2, v_3 and v_4 is a query motif, and its label weight is $w(\triangle_{234}) = 4$. The query motif support of edge (v_2, v_3) equals to*

12, since it's 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 $w_M(v_2, v_3) = 0.5 \times 12 + (1 - 0.5) = 6.5$. Figure 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.

3.2 Properties of LAM

In this section, we show that the LAM framework can effectively concentrate the PPR distribution from the seed node within the desired cluster. A concentrated PPR distribution can better reveal and distinguish the desired cluster from the rest of the graph [2]. For our theoretical analysis, we employ the stochastic block model [27] to generate random graphs of two clusters. As 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 of the 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 an undirected and unweighted 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 $p_{in} > p_{out}$. The nodes in cluster H_0 have label l_q and the 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 to show that LAM can effectively concentrate 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 N sufficiently large 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 recursively decided by the transition probabilities between H_0 and $H_{1\pm}$. 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 N sufficiently large 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 probability at least $1 - \delta$, where node u has in expectation $\bar{d}^j(u)$ degree of connection to H_j on G and in expectation $\bar{d}^j_M(u)$ degree of connection to H_j on G_M.

Proof. The proof on graph G can be achieved using Chernoff bounds [37] 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:

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

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

$$\Pr\Big(\exists u : d^j(u) \notin [(1-\gamma_2)\bar{d}^j(u), (1+\gamma_2)\bar{d}^j(u)]\Big) \leq O(Ne^{-N}) \quad (2)$$

According to the definition of G_M we have:

$$d^j_M(u) = \sum_{v \in H_j} \Big[(1-\lambda)A_{uv} + \lambda \cdot \Lambda_{uv} \cdot A_{uv}\Big] \quad (3)$$

where A is the adjacent matrix of G.

Combining formula 1,2,3 and union bounds, we 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 probability 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 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 probability 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 4 and 5 using a similar approach described in the first section of appendix for [27]. We omit the details here due to the space limit.

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:

$$R_t(H_i) \in \left[(1-\epsilon)\bar{R}_t(H_i), (1+\epsilon)\bar{R}_t(H_i)\right] \tag{4}$$

$$R^M_t(H_i) \in \left[(1-\epsilon)\bar{R}^M_t(H_i), (1+\epsilon)\bar{R}^M_t(H_i)\right] \tag{5}$$

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

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

with $\bar{R}_0(H_0) = \bar{R}^M_0(H_0) = 1, \bar{R}_0(H_{1\pm}) = \bar{R}^M_0(H_{1\pm}) = 0$.

With Equations 4 and 5, Lemma 2 is equivalent to: for any $0 < t \leq T$ and $0 < \lambda < 1$, $\bar{R}^M_t(H_0) > \bar{R}_t(H_0)$.

Let $p = \frac{p_{in}}{p_{out}} > 1$. As $\frac{\bar{d}^{1-}_M(H_j)}{\bar{D}_M(H_j)} = O(N^{-1})$ for $j \in \{0, 1+\}$, we can omit them given N is sufficiently large. Thus, $\bar{R}^M_t(H_{1-}) = \left(\frac{\bar{d}^{1-}_M(H_{1-})}{\bar{D}_M(H_{1-})}\right)^t \bar{R}^M_0(H_{1-}) = 0$. By further omitting quantities that are $O(N^{-1})$, we can derive:

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

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:

$$\bar{R}^M_t(H_0) = a\bar{R}^M_{t-1}(H_0) + b$$

which implies:

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

We can observe that $\bar{R}^M_t(H_0)$ 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:

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

for which we complete the proof. □

Note that for our analysis, we do not require any distribution on the lengths of random walks. Thus, our analysis is also applicable for other random walk ranking models such as the heat-kernel pagerank [26] and the inversed pagerank [30].

4 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 on labeled graphs by optimizing the proposed cluster metric. The complexity analysis of the index-free algorithm is presented in Section 4.3.

4.1 Cluster Metric based on LAM

As presented in Section 2.1, 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 $\phi_M(H)$ for measuring structure density as LAM effectively increases structure density within the desired cluster as shown in Section 3.2. Meanwhile, we allow $\rho(H, L_q)$ to be any well-thought function that measures the label density of H. One can use $\rho(H, L_q)$ from the existing works on community search [14, 23]. For example, [55] defines $\rho(H, L_q) = \rho_1(H, L_q) = \sum_{l \in L_q} \frac{|V(l) \cap V(H)|^2}{|V(H)|}$. To ease the presentation on subsequent examples and analysis, we use a simple and intuitive formulation $\rho(H, L_q) = \rho_2(H, L_q) = \frac{\sum_u |L_q \cap L(u)|}{|H|}$, which measures the average number of matched query labels in H. Choosing the best $\rho(H, L_q)$ is orthogonal to the contribution of this work and we conduct experiments to show that our general formulation can effectively find labeled clusters for any well-thought $\rho(H, L_q)$.

Next, we show that optimizing $f_q(H)$ is generally NP-hard.

Theorem 2. *There exists a function $\rho(H, L_q)$ such that optimizing the cluster metric $f_q(H) = \frac{\rho(H, L_q)}{\phi_M(H)}$ is NP-hard.*

Proof. The proof is trivial when considering the constant function $\rho(H, L_q) = 1$, i.e., the label information is not considered by the metric. Then, maximizing $f_q(H)$ is equivalent to minimizing $\phi_M(H)$ on G_M, which is a well-known NP-hard problem [44]. □

Algorithm 1: LAM Clustering

Input: Graph G, the seed node s and the set of query labels L_q, error tolerance ϵ
Output: The subgraph H^* with maximum $f_q(\cdot)$

1 compute PPR $\pi_M(s)$ on G_M with error tolerance ϵ;
2 $\bar{\pi}_M(s) \leftarrow D^{-1}\pi_M(s)$ and D is degree matrix of G;
3 sort nodes in G by descending $\bar{\pi}_M(s)$ and u_i denotes the i-th node after sorting;
4 $H \leftarrow \arg\min_{H_i} \phi_M(H_i)$, where $H_i = \{u_1, ..., u_i\}$;
5 $\rho \leftarrow 0$;
6 **while** $\rho < \rho(H, L_q)$ **do**
7 $\quad \rho \leftarrow \rho(H, L_q)$;
8 $\quad v_r \leftarrow \arg\min_{u \in H} \text{dep}(u, H)$;
9 $\quad$ remove v_r from H;
10 **end**
11 $H^* \leftarrow H \cup \{v_r\}$, where v_r is the last removed node;
12 **return** H^*

4.2 Index-free Two-stage Algorithm

As maximizing the cluster metric $f_q(H)$ is NP-hard, we devise a two-stage heuristic to maximize $f_q(H)$. Note that $f_q(H)$ 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 $\phi_M(H)$. Then, 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 label information in the processing. In contrast, maximizing $\rho(H, L_q)$ first may 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. 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 Algorithm 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 details of the algorithm as follows.

Stage I: minimizing $\phi_M(H)$ (Lines 1-4). We adopt the sweep algorithm [3] for minimizing $\phi_M(H)$. The sweep algorithm first calculates the PPR distribution from the seed node, and then applies a "sweep" to identify a cluster with a low LAM conductance score. More specifically, the PPR value for each node u in the LAM graph G_M from the seed s, denoted as $\pi_M(s, u)$, is computed (Line 1) and is then normalized with its degree $d(u)$ (Line 2). Subsequently, the nodes are sorted by the normalized PPR values in a descending order (Line 3). Finally, a "sweep" operation is applied to examine each *prefix* 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 4).

Stage II: maximizing $\rho(H, L_q)$ (Lines 5-11). 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* $\text{dep}(u, H)$, for each node u in H. This density perturbation $\text{dep}(u, H)$ 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 $\text{dep}(v_r, H)$, is iteratively removed from H until the $\rho(H, L_q)$ cannot be increased anymore.

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

$$\text{dep}(u, H) = |L(u) \cap L_q| \cdot \frac{\phi_M(H - \{u\})}{\phi_M(H)}$$

The first factor of $\text{dep}(u, H)$ measures the impact of removing u to the label density $\rho(H, L_q)$; whereas the second factor is the ratio between the LAM conductance before and after removing u.

Implementation Details. A naïve way to compute the PPR distribution over G_M is to first construct the entire G_M explicitly and then apply any PPR algorithm on G_M. However, constructing the entire G_M requires finding all query motifs (i.e. triangles) in G, which is prohibitively expensive. Hence, we extend the LocalPush algorithm [3], which only requires accessing nodes around the seed node s locally, to compute the approximate PPR distribution without constructing the entire G_M. The LocalPush algorithm computes $\pi_M(s)$ by accessing at most $O(\frac{1}{\epsilon})$ edges [3]. Hence, we can efficiently obtain a good initial cluster with the LocalPush extension.

Moreover, stage II requires computing the $\text{dep}(u, H) \propto \phi_M(H - \{u\})$ for all $u \in H$ in each iteration.

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

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 $\text{dep}(u, H)$ 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 $\text{dep}(u, H)$ 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 $d_{in}(u, H)$ 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.

4.3 Complexity Analysis

This section presents the worst-case time complexity of stages I and II of our algorithm respectively. The following lemma gives the time complexity of stage I. 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.

Lemma 3. *The time complexity of stage I is $O(nd_{max}L_{max}+n\log n)$, where n is the number of nodes in G.*

Proof. We first give the time complexity of obtaining the PPR distribution over G_M. The key point here is to calculate the complexity for computing $sup(v_i, v_j, L_q)$ for an edge (v_i, v_j). The support value $sup(v_i, v_j, L_q)$ 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 $u \in N(v_i) \cap N(v_j)$, the algorithm computes $|L(u) \cap L_q|$ by scanning $L(u)$, which can be completed in $O(|L(u)|) = O(L_{max})$. In summary, the time complexity for computing $sup(v_i, v_j, L_q)$ is $O(d_{max} \cdot L_{max})$. Furthermore, we implement the algorithm with LocalPush, which only requires query motif support value of $O(\frac{1}{\epsilon})$ edges. Thus, the time complexity for computing PPR distribution over G_M is $O(d_{max} \cdot L_{max}) \cdot O(\frac{1}{\epsilon}) = O(\frac{d_{max} \cdot L_{max}}{\epsilon})$. Then, the nodes in H are sorted to obtain the prefix nodes set with the smallest LAM conductance, which takes $O(n\log n)$ time.

A common practice is to set $\epsilon = O(\frac{1}{n})$. Thus, the time complexity is bounded by $O(nd_{max}L_{max} + n\log n)$. □

The following lemma gives the time complexity of stage II.

Lemma 4. *The time complexity of stage II is bounded by $O(|H|^2 + \mathrm{vol}(H))$, where H is the output cluster of stage I.*

Proof. 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 $O(\mathrm{vol}(H))$ time. In any iteration, the computation of $\mathrm{dep}(u, H)$ for each node u takes $O(|H|)$ time. As 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 $O(|H|^2 + \mathrm{vol}(H))$. □

Lemma 4 shows the quadratic complexity of stage II w.r.t. the output H by stage I. To ensure efficient execution, H should minimize the number of unpromising nodes. An irrelevant node can be included in H if it has larger normalized PPR value than any node in the desired cluster. Thus, by concentrating the PPR value within the desired cluster, LAM can potentially return a better coarse-grained cluster in stage I, as stated in Theorem 1.

We also evaluate the average size of the coarse-grained cluster H returned by stage I of our algorithm on the datasets used in our experiments, as shown in Figure 3. When the LAM framework is applied, the average size of H is consistently smaller than 2500 on all datasets. For the unweighted conductance, the average size of H can reach 60000 on the youtube dataset. Thus, the algorithm based on the unweighted conductance cannot finish within one hour for such huge H, which shows a surprising advantage of adopting the LAM conductance. To ensure the algorithm based on the unweighted conductance to terminate within a reasonable time, we return H with the smallest conductance among the first 1000 prefix node sets, which achieves good performance empirically.

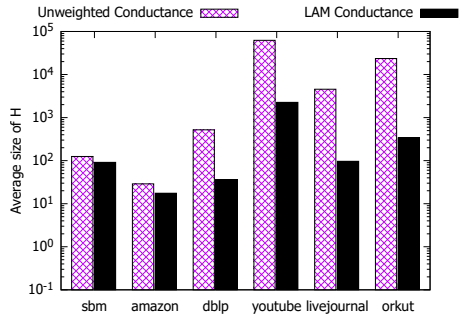

Figure 3: The average size of the coarse-grained cluster H returned by stage I.

5 EXPERIMENTS

We evaluate the LAM framework with the two-stage algorithm through extensive experiments on both real-world and sythetic datasets. Section 5.1 describes the setup. Subsequent subsections are presented to answer the following research questions:

- Is parameter tuning made easy for LAM (Section 5.2)?
- Can LAM-based methods outperform the state-of-the-art solutions for identifying the desired cluster in both real and synthetic datasets (Section 5.3)?
- Is LAM robust to label noise (Section 5.4)?
- Is the two-stage algorithm efficient and scalable enough to handle large graphs (Section 5.5)?
- Can queries with different labels reveal diverse yet cohesive clusters (Section 5.6)?

5.1 Experimental Setup

Datasets. We collect five real-world networks with ground-truth clusters from SNAP [29]: amazon, dblp, youtube, livejournal and orkut. We also include synthetic random graphs generated from the stochastic block model $P(N, k, p_{in}, p_{out})$, 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. We set $\mu = 0.9$ by default. Network statistics of the datasets are reported in Table 2.

The above datasets do not contain label information. Although there exist many real-world labeled graph datasets with ground-truth clusters, the ground-truth information is *not* suitable for evaluating labeled-aware query for local clustering. This is because the ground-truth clusters do not change regardless of the provided query labels. In Section 5.6, we conduct a case study on a labeled graph dataset and demonstrate that diverse clusters are extracted for different query labels on the same labeled ground-truth cluster. Hence, the ground-truth information in labeled graphs is not query-specific and is not applicable for our evaluation. To this end, we follow the existing works on labeled-aware community search [14, 23] which generate the node labels in unlabeled graphs based

Table 2: Dataset Statistics. The last column Φ gives the average unweighted conductance of ground-truth clusters in the network. `sbm` is the network generated by the stochastic block model with $\mu = 0.9$.

Network	$\|V\|$	$\|E\|$	d_{max}	Φ
sbm	500	125K	288	0.9
amazon	335K	926K	549	0.061
dblp	317K	1M	342	0.414
youtube	1.1M	3M	28754	0.896
livejournal	4M	35M	14815	0.385
orkut	3.1M	117M	33313	0.732

on the ground-truth clusters and issue labeled queries to validate if we can extract the ground-truth clusters.

Label Generation. We follow the setup used in [23] to augment labels for all datasets. We generate a label set consisting of $|\mathcal{L}| = \eta \cdot |V|$ distinct labels for each dataset. We set $\eta = 0.0001$ by default for real-world networks and set $\eta = 0.02$ by default for synthetic networks generated by the stochastic block model. For each ground-truth cluster, we randomly select 3 labels from $\mathcal{L}$, 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.

Query Generation. We generate one query for each ground-truth cluster by randomly selecting one node from the cluster as the seed node, and choosing the corresponding 3 labels assigned to this ground-truth cluster as the query labels. We randomly select the seed node to avoid the influence of different seeding algorithms. Note that seed selection is orthogonal to our work.

Compared Methods. We compared the proposed methods with state-of-the-art methods on label-aware community search.

- ATC [23] returns a (k, d)-truss community containing the seed node that maximizes a specifically designed label score.
- ACQ [14] 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.
- TEC1 uses the unweighted conductance with $\rho(H, L_q) = \rho_1(H, L_q)$ and employs our two-stage algorithm to extract the cluster.
- LAM1 is our proposed approach with the LAM conductance and $\rho(H, L_q) = \rho_1(H, L_q)$ used in $f_q(H)$.
- TEC2 is a variant of TEC1 with $\rho(H, L_q) = \rho_2(H, L_q)$.
- LAM2 is a variant of LAM1 with $\rho(H, L_q) = \rho_2(H, L_q)$.

We use two different definitions for label density $\rho(H, L_q)$ to show that our LAM framework can improve the effectiveness of the two-stage algorithm with any well-thought $\rho(H, L_q)$.

Note that for ACQ and ATC, suitable hyperparameters k and d have to be assigned for each query. We follow the setup used in [23] and set the 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 also set as the same in [23].

Parameter Setup. The parameters used in the experiments and their default values are summarized in Table 3.

Table 3: 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
p_{err}	The probability of label noise within ground-truth clusters.	0.0
λ	The hyperparameter in LAM.	0.4

Evaluation Metric. In the experiments, 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 3.6 GHz 72-core CPU and 256 GB memory running Ubuntu 18.04. We use the original implementation of ATC (implemented with C++) and ACQ (implemented with Java) provided by the authors. Other algorithms are implemented with C++ and executed with a single thread.

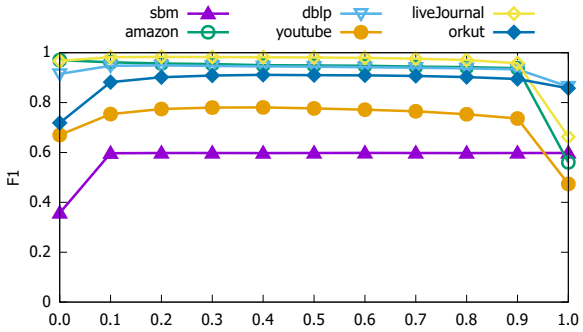

Figure 4: Varying λ for LAM2.

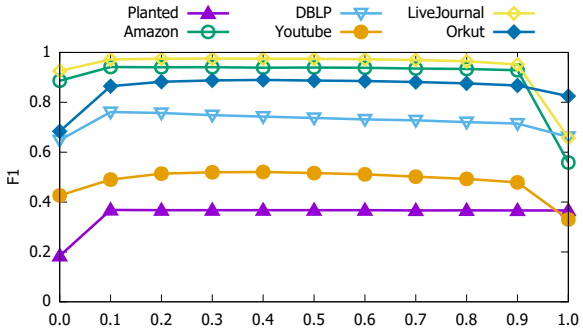

Figure 5: Varying λ for LAM1.

5.2 Parameter Tuning for λ in LAM

We vary the hyperparameter λ used in the LAM framework and report the F1 scores of LAM2 and LAM1 in Figure 4 and Figure 5 respectively.

We have the following observations for LAM2. First, the performance is relatively stable when λ is from 0.1 to 0.9, 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

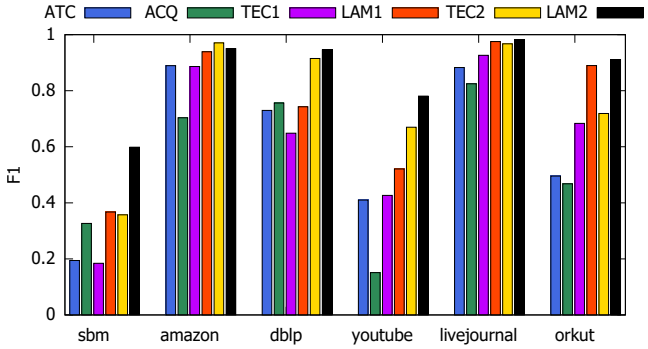

Figure 6: Evaluation on the real networks.

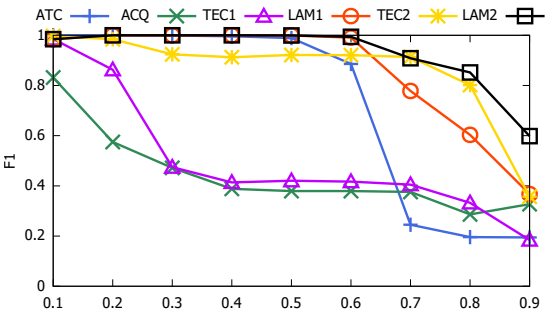

Figure 7: Evaluation on the synthetic networks.

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 LAM2 slightly decrease on the amazon dataset, while first increase and then decrease on the youtube and orkut datasets. With a larger λ, the LAM framework places more emphasis on the query motif, which results in the PPR distribution to be more concentrated around the seed node. Hence, on datasets amazon with well-structured ground-truth clusters (i.e., small average unweighted conductance as shown in Table 2), 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 once λ is larger than 0.1. On the youtube and orkut datasets with ill-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. Similarly, on datasets dblp and livejournal with moderately-structured ground-truth clusters, the F1 score achieve its peak value at $\lambda = 0.2$, which is smaller than the best λ on youtube and orkut. We can make similar observations in Figure 5 for LAM1.

According to the observations in Figure 4 and Figure 5, 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 achieves a balance between the original graph and the adjusted edge weights by the query motif instances. Compared with the topology-driven community search models, which set different k values even for different queries, tuning the hyperparameter λ is easier for non-expert users.

5.3 Effectiveness Evaluation

We evaluate the overall effectiveness of our proposed methods based on the LAM framework (LAM1 and LAM2) against both the topology-driven community search methods (ATC and ACQ) and the unweighted conductance methods (TEC1 and TEC2). Figure 6 reports the F1 scores of the compared methods on each dataset.

Overall, we can make two main observations from the results.

First, across all datasets, the algorithms based on LAM framework (especially LAM2) outperform the topology-driven community search methods (ATC and ACQ). On the amazon and livejournal datasets, the gap between LAM based methods and ATC, ACQ is relatively small. The reason is that these two datasets have well-structured ground-truth clusters (i.e., low averaged conductance), it is easy for the baselines to extract the ground-truth clusters as these clusters have better topological features for extraction. In contrast, on the sbm, dblp, youtube and orkut datasets, the gap is much more significant. For example, LAM2 achieves 90.5% and 84% relative improvements over ATC on the youtube and orkut datasets, respectively. The reason is that ATC and ACQ are highly dependent on the structure of the ground-truth clusters. However, when the topological features are not "clear" (i.e., the conductance scores are high), the baselines show significantly inferior F1 scores. The results have validated that LAM is more robust in terms of handling structure noises in ground-truth clusters than the baselines.

Second, both LAM1 and LAM2 outperform the corresponding TEC1 and TEC2 in most of the cases, which shows that our LAM framework can indeed boost the effectiveness of the proposed two-stage algorithm regardless of different choices for $\rho(H, L_q)$. Note that the effectiveness of LAM2 is slightly worse than TEC2 on amazon. This is still due to the well-structured ground-truth clusters of amazon, which makes it easy to extract the clusters out solely based on unweighted conductance.

We further use the synthetic networks with different mixing ratio μ on the sbm dataset to evaluate all compared methods. A larger μ implies the number of edges between clusters increases which makes the clusters harder to extract. The experimental result is shown in Figure 7. Overall, we find that LAM1 and LAM2 consistently outperforms the baselines with various values of μ. For the baseline methods, when μ is varied from 0.1 to 0.6, all the methods except ACQ and TEC1 can achieve satisfactory F1 scores, while ACQ and TEC1 experiences sharp drops in F1 scores when μ is varied from 0.1 to 0.3. When μ is increased from 0.6 to 0.7, the F1 score of ATC incurs a very sharp drop. The scores of TEC2 and LAM2 keep relatively stable until $\mu = 0.8$. When μ is increased from 0.8 to 0.9, TEC2 incurs a sharper drop than LAM2.

In summary, we obtain the ranking of the compared methods in terms of handling structure noises of ground-truth clusters: ACQ <TEC1 <ATC <LAM1 <TEC2 <LAM2.

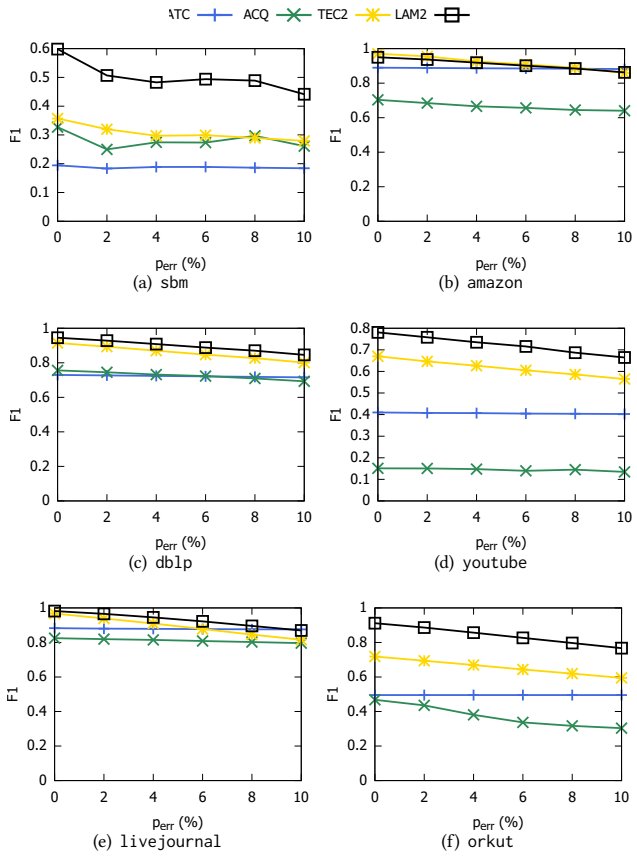

Figure 8: Varying p_{err} for LAM2.

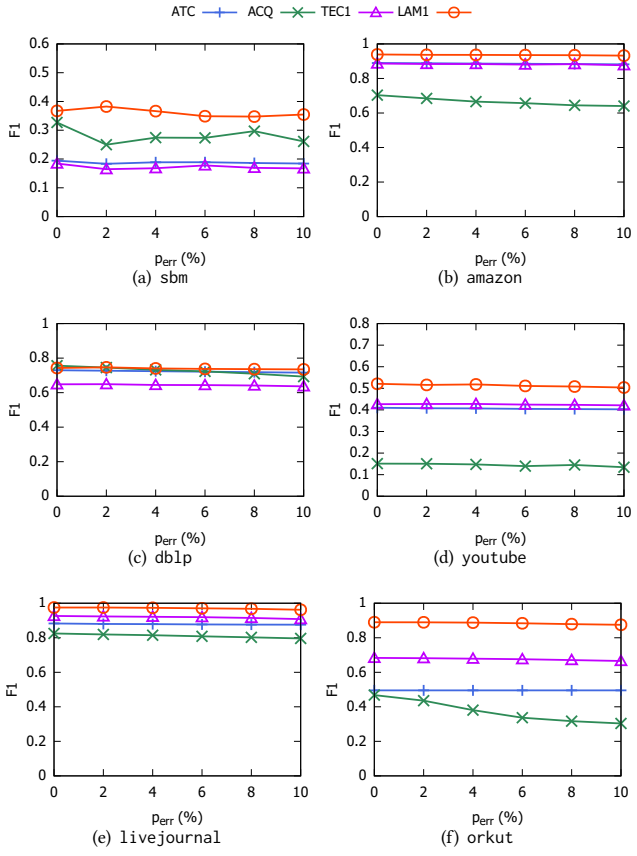

Figure 9: Varying p_{err} for LAM1.

5.4 Label Noise Robustness

We vary parameters p_{err} and η of label assignments to evaluate the robustness of all compared methods to label noises.

Varying p_{err}. The parameter p_{err} measures the label noise within each ground-truth cluster, i.e., the probability that a node within the ground-truth cluster is not assigned with the desired labels of the queries. We vary p_{err} from 2% to 10%. Figure 8 and Figure 9 compares LAM2 and LAM1 with baselines respectively when p_{err} is varied. The scores drop in general as there are more noises introduced in the label information.

Our proposed LAM2 and LAM1 algorithms still achieve consistently better results than those produced by the topology-driven community search methods (ATC and ACQ) as well as the corresponding unweighted conductance methods (TEC2 and TEC1) over all datasets.

We also see that LAM2 and TEC2 are relatively more sensitive to label noise within the ground-truth clusters. This is because LAM2 and TEC2 extensively leverages the label information to trim nodes in the second-stage. The noise in the label information will affect the quality of the trimming process.

Varying η. The parameter η measures the label noise 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. A smaller η means a larger label noise. 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 10 and Figure 11 report the F1 scores with varying η. As in the previous experiments, LAM2 and LAM1 are consistently better than other baselines.

5.5 Efficiency and Scalability

In this section, we compare the topology-driven approach (ATC) and our conductance-driven approach (LAM2) for their efficiency and scalability. The purpose is to demonstrate the efficacy of LAM2's index-free algorithm. ACQ and TEC2 are omitted because they produce inferior cluster qualities than ATC and LAM2, respectively.

Time Consumption. Figure 12(a) compares the running time of our proposed algorithm LAM2 and ATC. Note that ATC needs to construct indices before processing queries. Thus, we compare the running time of 1000 queries for LAM2 with the running time of 1000 queries plus time of index construction for ATC.

We can find that on small datasets with well-structured ground-truth clusters such as `amazon` (with average unweighted conductance 0.06), ATC is more efficient than LAM2. However, in large datasets or datasets with ill-structured ground-truth clusters, LAM2 is more efficient than ATC. For example, on the large `livejournal` dataset and with well-structured clusters, although the query processing of ATC is very efficient based on indices, index construction dominates the time consumption (73%) and makes ATC as a whole less efficient than LAM2. On the datasets such as `sbm`, `youtube` and

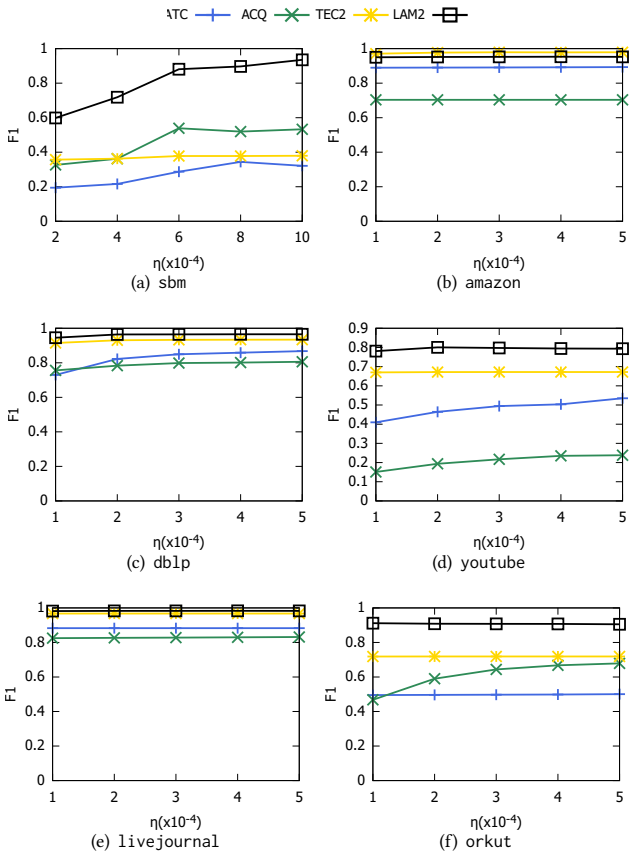

Figure 10: Varying η. for LAM2

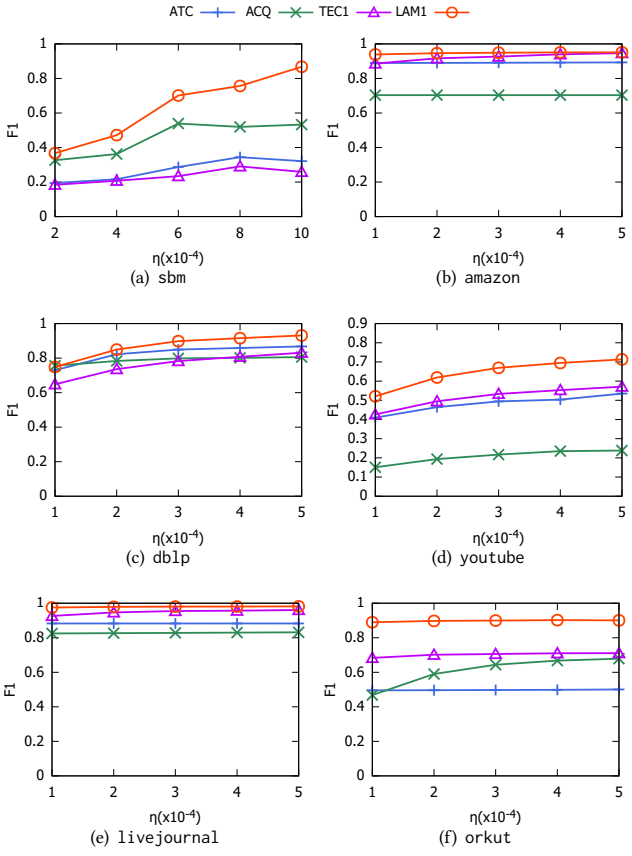

Figure 11: Varying η. for LAM1

`orkut`, LAM2 can even process queries more efficient than ATC. In summary, our proposed method is at least competitive with ATC in terms of time consumption.

Memory Consumption. Figure 12(b) reports the memory consumption of LAM2 and ATC. We can find that due to large indices required by ATC, it consumes significantly more memory than index-free LAM2. The gap between memory consumption of two methods scales with the size of the network. For small networks `sbm`, `dblp`, `youtube`, ATC requires 3 to 4 times memory than LAM2. For larger networks `livejournal`, `orkut`, ATC requires up to 10 times memory than LAM2. This implies that LAM2 is much more efficient than ATC in terms of memory consumption.

Scalability. We further compare the scalability of LAM2 with ATC using a large network `friendster` from SNAP [29] with more than 66 million nodes and 1.8 billion edges. ATC cannot handle queries from `friendster` since it fails to construct the indices with 256GB memory on our machine. In contrast, our method can handle queries from `friendster` with 60GB memory. We randomly select 100 queries in `friendster` and it takes 223 seconds on average to process one query for LAM2. Furthermore, the performance can be easily accelerated by parallel processing [5]. Thereby, LAM2 is more scalable than topology-based ATC to handle large networks.

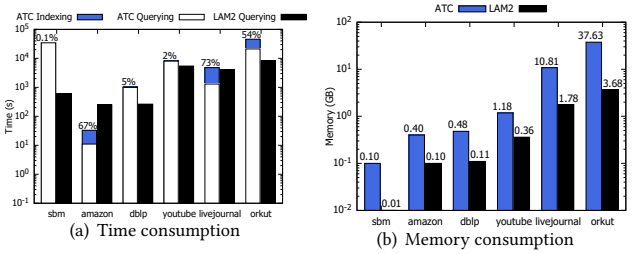

Figure 12: Time and memory consumption: the percentages over the column of ATC in (a) represents the ratio of time for constructing indices to the total time consumed.

5.6 A Case Study on Facebook Labeled Graph

In this subsection, we further validate that LAM2 can extract label-aware and cohesive clusters from real-world labeled graphs. We conduct a case study on a `facebook` dataset from SNAP [29]. Each node in the `facebook` dataset has labels representing features like political stand and education degree. The label/feature values are anonymized to numbers for privacy concern.

We study all ego-networks in the `facebook` dataset and the statistics of the dataset is presented in Table 4. To verify that LAM2 can detect label-aware clusters, we issue queries with different query labels and show that LAM2 can get diverse results. For each ground-truth cluster in one ego-network, we select top-2 representative labels (l_1 and l_2 respectively), i.e. the labels that maximize the ratio of label frequency within vs. outside the cluster. For each

Table 4: Statistics for ego-networks in facebook. d_{max} denote the maximum node degree in the ego-network, and L_{max} denote the maximum number of labels attached to a node.

Network	$\|V\|$	$\|E\|$	d_{max}	L_{max}
fb0	347	2519	77	32
fb107	1045	26749	253	36
fb348	227	3192	99	23
fb414	755	1693	57	20
fb686	170	1656	77	17
fb698	66	270	29	14
fb1684	792	14024	136	22
fb1912	755	30025	293	41
fb3437	547	4813	107	33
fb3980	59	146	18	15

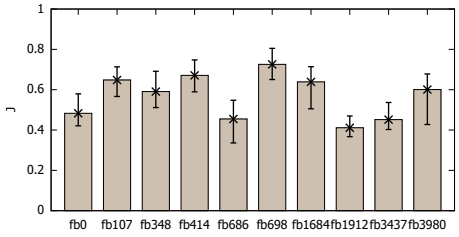

Figure 13: Average Jaccard similarity J between two clusters obtained by different query labels.

representative label l, we issue a clustering query with label set $\{l\}$ and a random node in the ground-truth cluster. Figure 13 reports the average Jaccard similarity between results obtained by the two queries on each ego-network in facebook dataset. The results show that LAM2 can indeed extract different label-aware clusters wrt. different query labels.

We further look more closely into the clusters returned. In Figure 14, we visualize two clusters extracted by different query labels l_1 and l_2 from ego-networks fb414 and fb1912. In fb414, we can intuitively see that both two clusters extracted for query label l_1 and l_2 have high structure density. For 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 with nodes that are attached with label l_1 (nodes colored in red). Note that several nodes with l_1 are also excluded by LAM2 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.

6 CONCLUSION

In this paper, we introduced the problem of local clustering on labeled graphs. We observed that some existing topology-driven labeled community search approaches can be adapted to our problem. However, these methods suffers from the requirment of strict topology constraints and prohibitively large indices. We thus proposed

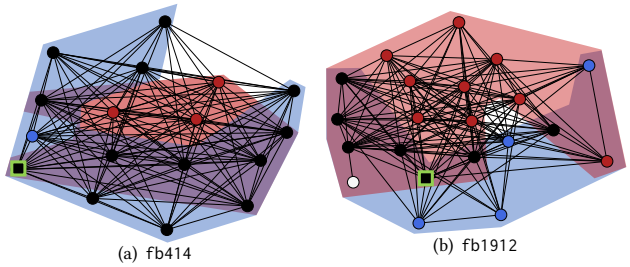

Figure 14: Examples of clusters returned with different query labels. In each ego-network of facebook, two queries are issued from the highlighted seed in the green square with query label l_1 and l_2 respectively. The subgraph shaded with red(blue) is the cluster returned for label $l_1(l_2)$. The color 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.

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 existing community search approaches, our proposed 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] Esra Akbas and Peixiang Zhao. 2017. Truss-based community search: a truss-equivalence based indexing approach. *Proceedings of the VLDB Endowment* 10, 11 (2017), 1298–1309.
[2] Reid Andersen and Fan Chung. 2007. Detecting sharp drops in PageRank and a simplified local partitioning algorithm. In *International Conference on Theory and Applications of Models of Computation*. Springer, 1–12.
[3] Reid Andersen, Fan Chung, and Kevin Lang. 2006. Local graph partitioning using pagerank vectors. In *the 2006 IEEE FOCS Annual Symposium on Foundations of Computer Science*. 475–486.
[4] Alex Arenas, Alberto Fernandez, Santo Fortunato, and Sergio Gomez. 2008. Motif-based communities in complex networks. *Journal of Physics A: Mathematical and Theoretical* 41, 22 (2008), 224001.
[5] Bahman Bahmani, Kaushik Chakrabarti, and Dong Xin. 2011. Fast personalized pagerank on mapreduce. In *Proceedings of the 2011 ACM SIGMOD international conference on management of data*. 973–984.
[6] Nicola Barbieri, Francesco Bonchi, Edoardo Galimberti, and Francesco Gullo. 2015. Efficient and effective community search. *Data mining and knowledge discovery* 29, 5 (2015), 1406–1433.
[7] Lijun Chang, Xuemin Lin, Lu Qin, Jeffrey Xu Yu, and Wenjie Zhang. 2015. Index-based optimal algorithms for computing steiner components with maximum connectivity. In *Proceedings of the 2015 ACM SIGMOD international conference on management of data*. 459–474.
[8] Lu Chen, Chengfei Liu, Kewen Liao, Jianxin Li, and Rui Zhou. 2019. Contextual community search over large social networks. In *the 2019 IEEE ICDE international conference on data engineering*. 88–99.
[9] Lingyang Chu, Zhefeng Wang, Jian Pei, Yanyan Zhang, Yu Yang, and Enhong Chen. 2019. Finding theme communities from database networks. *Proceedings of the VLDB Endowment* 12, 10 (2019), 1071–1084.
[10] Aaron Clauset. 2005. Finding local community structure in networks. *Physical review E* 72, 2 (2005), 026132.
[11] Wanyun Cui, Yanghua Xiao, Haixun Wang, Yiqi Lu, and Wei Wang. 2013. Online search of overlapping communities. In *Proceedings of the 2013 ACM SIGMOD international conference on management of data*. 277–288.
[12] Wanyun Cui, Yanghua Xiao, Haixun Wang, and Wei Wang. 2014. Local search of communities in large graphs. In *Proceedings of the 2014 ACM SIGMOD international conference on management of data*. 991–1002.
[13] Elizabeth M Daly and Mads Haahr. 2007. Social network analysis for routing in disconnected delay-tolerant manets. In *Proceedings of the 2007 ACM MOBIHOC international symposium on mobile ad hoc networking and computing*. 32–40.

[14] Yixiang Fang, Reynold Cheng, Siqiang Luo, and Jiafeng Hu. 2016. Effective community search for large attributed graphs. *Proceedings of the VLDB Endowment* 9, 12 (2016), 1233–1244.

[15] Yixiang Fang, Zhongran Wang, Reynold Cheng, Hongzhi Wang, and Jiafeng Hu. 2018. Effective and efficient community search over large directed graphs. *IEEE TKDE transactions on knowledge and data engineering* 31, 11 (2018), 2093–2107.

[16] Santo Fortunato. 2010. Community detection in graphs. *Physics reports* 486, 3-5 (2010), 75–174.

[17] Michelle Girvan and Mark EJ Newman. 2002. Community structure in social and biological networks. *Proceedings of the national academy of sciences* 99, 12 (2002), 7821–7826.

[18] David F Gleich and C Seshadhri. 2012. Vertex neighborhoods, low conductance cuts, and good seeds for local community methods. In *Proceedings of the 2012 ACM SIGKDD international conference on knowledge discovery and data mining*. 597–605.

[19] Jiafeng Hu, Xiaowei Wu, Reynold Cheng, Siqiang Luo, and Yixiang Fang. 2017. On minimal steiner maximum-connected subgraph queries. *IEEE TKDE transactions on knowledge and data engineering* 29, 11 (2017), 2455–2469.

[20] Ling Huang, Chang-Dong Wang, and Hong-Yang Chao. 2018. A harmonic motif modularity approach for multi-layer network community detection. In *the 2018 IEEE ICDM international conference on data mining*. 1043–1048.

[21] Xin Huang, Hong Cheng, Lu Qin, Wentao Tian, and Jeffrey Xu Yu. 2014. Querying k-truss community in large and dynamic graphs. In *Proceedings of the 2014 ACM SIGMOD international conference on management of data*. 1311–1322.

[22] Xin Huang, Hong Cheng, and Jeffrey Xu Yu. 2015. Dense community detection in multi-valued attributed networks. *Information Sciences* 314 (2015), 77–99.

[23] Xin Huang and Laks VS Lakshmanan. 2017. Attribute-driven community search. *Proceedings of the VLDB Endowment* 10, 9 (2017), 949–960.

[24] Xin Huang, Laks VS Lakshmanan, Jeffrey Xu Yu, and Hong Cheng. 2015. Approximate closest community search in networks. *Proceedings of the VLDB Endowment* 9, 4 (2015), 276–287.

[25] Lucas GS Jeub, Prakash Balachandran, Mason A Porter, Peter J Mucha, and Michael W Mahoney. 2015. Think locally, act locally: Detection of small, medium-sized, and large communities in large networks. *Physical Review E* 91, 1 (2015), 012821.

[26] Kyle Kloster and David F Gleich. 2014. Heat kernel based community detection. In *Proceedings of the 2014 ACM SIGKDD international conference on knowledge discovery and data mining*. 1386–1395.

[27] Isabel M Kloumann, Johan Ugander, and Jon Kleinberg. 2017. Block models and personalized PageRank. *Proceedings of the national academy of sciences* 114, 1 (2017), 33–38.

[28] Kevin J Lang and Reid Andersen. 2007. Finding dense and isolated submarkets in a sponsored search spending graph. In *Proceedings of the 2007 ACM CIKM conference on conference on information and knowledge management*. 613–622.

[29] Jure Leskovec and Andrej Krevl. 2014. SNAP Datasets: Stanford Large Network Dataset Collection. http://snap.stanford.edu/data.

[30] Pan Li, I Chien, and Olgica Milenkovic. 2019. Optimizing generalized pagerank methods for seed-expansion community detection. In *the 2019 NeurIPS advances in neural information processing systems*. 11710–11721.

[31] Pei-Zhen Li, Ling Huang, Chang-Dong Wang, and Jian-Huang Lai. 2019. EdMot: An edge enhancement approach for motif-aware community detection. In *Proceedings of the 2019 ACM SIGKDD international conference on knowledge discovery and data mining*. 479–487.

[32] Yixuan Li, Kun He, Kyle Kloster, David Bindel, and John Hopcroft. 2018. Local spectral clustering for overlapping community detection. *ACM TKDD transactions on knowledge discovery from data* 12, 2 (2018), 1–27.

[33] Qing Liu, Minjun Zhao, Xin Huang, Jianliang Xu, and Yunjun Gao. 2020. Truss-based community search over large directed graphs. In *Proceedings of the 2020 ACM SIGMOD international conference on management of data*. 2183–2197.

[34] Qing Liu, Yifan Zhu, Minjun Zhao, Xin Huang, Jianliang Xu, and Yunjun Gao. 2020. VAC: Vertex-Centric Attributed Community Search. In *the 2020 IEEE ICDE international conference on data engineering*. IEEE, 937–948.

[35] Feng Luo, James Z Wang, and Eric Promislow. 2006. Exploring local community structures in large networks. In *the 2006 IEEE/WIC/ACM WI international conference on web intelligence*. 233–239.

[36] Farnaz Moradi, Tomas Olovsson, and Philippas Tsigas. 2014. A local seed selection algorithm for overlapping community detection. In *the 2014 IEEE/ACM ASONAM international conference on advances in social networks analysis and mining*. 1–8.

[37] Rajeev Motwani and Prabhakar Raghavan. 1995. *Randomized algorithms*. Cambridge university press.

[38] Roger Pearce. 2017. Triangle counting for scale-free graphs at scale in distributed memory. In *the 2017 IEEE HPEC high performance extreme computing conference*. 1–4.

[39] Yiye Ruan, David Fuhry, and Srinivasan Parthasarathy. 2013. Efficient community detection in large networks using content and links. In *Proceedings of the 2013 ACM WWW international conference on world wide web*. 1089–1098.

[40] Barna Saha, Allison Hoch, Samir Khuller, Louiqa Raschid, and Xiao-Ning Zhang. 2010. Dense subgraphs with restrictions and applications to gene annotation graphs. In *Annual International Conference on Research in Computational Molecular Biology*. Springer, 456–472.

[41] Comandur Seshadhri, Tamara G Kolda, and Ali Pinar. 2012. Community structure and scale-free collections of Erdős-Rényi graphs. *Physical Review E* 85, 5 (2012), 056109.

[42] Jianbo Shi and Jitendra Malik. 2000. Normalized cuts and image segmentation. *IEEE TPAMI transactions on pattern analysis and machine intelligence* 22, 8 (2000), 888–905.

[43] Julian Shun and Kanat Tangwongsan. 2015. Multicore triangle computations without tuning. In *the 2015 IEEE ICDE international conference on data engineering*. 149–160.

[44] Jiří Šíma and Satu Elisa Schaeffer. 2006. On the NP-completeness of some graph cluster measures. In *International Conference on Current Trends in Theory and Practice of Computer Science*. Springer, 530–537.

[45] Mauro Sozio and Aristides Gionis. 2010. The community-search problem and how to plan a successful cocktail party. In *Proceedings of the 2010 ACM SIGKDD international conference on knowledge discovery and data mining*. 939–948.

[46] Daniel A Spielman and Shang-Hua Teng. 2013. A local clustering algorithm for massive graphs and its application to nearly linear time graph partitioning. *SIAM Journal on computing* 42, 1 (2013), 1–26.

[47] Charalampos Tsourakakis, Francesco Bonchi, Aristides Gionis, Francesco Gullo, and Maria Tsiarli. 2013. Denser than the densest subgraph: extracting optimal quasi-cliques with quality guarantees. In *Proceedings of the 2013 ACM SIGKDD international conference on knowledge discovery and data mining*. 104–112.

[48] Charalampos E Tsourakakis, Jakub Pachocki, and Michael Mitzenmacher. 2017. Scalable motif-aware graph clustering. In *Proceedings of the 2017 ACM WWW international conference on world wide web*. 1451–1460.

[49] Twan Van Laarhoven and Elena Marchiori. 2016. Local network community detection with continuous optimization of conductance and weighted kernel k-means. *The Journal of Machine Learning Research* 17, 1 (2016), 5148–5175.

[50] Yue Wang, Xun Jian, Zhenhua Yang, and Jia Li. 2017. Query optimal k-plex based community in graphs. *Data Science and Engineering* 2, 4 (2017), 257–273.

[51] Duncan J Watts and Steven H Strogatz. 1998. Collective dynamics of 'small-world'networks. *Nature* 393, 6684 (1998), 440–442.

[52] Yubao Wu, Ruoming Jin, Jing Li, and Xiang Zhang. 2015. Robust local community detection: on free rider effect and its elimination. *Proceedings of the VLDB Endowment* 8, 7 (2015), 798–809.

[53] Jaewon Yang, Julian McAuley, and Jure Leskovec. 2013. Community detection in networks with node attributes. In *the 2013 IEEE ICDM international conference on data mining*. 1151–1156.

[54] Hao Yin, Austin R Benson, and Jure Leskovec. 2019. The local closure coefficient: A new perspective on network clustering. In *Proceedings of the 2019 ACM WSDM international conference on web search and data mining*. 303–311.

[55] Hao Yin, Austin R Benson, Jure Leskovec, and David F Gleich. 2017. Local higher-order graph clustering. In *Proceedings of the 2017 ACM SIGKDD international conference on knowledge discovery and data mining*. 555–564.

[56] Long Yuan, Lu Qin, Wenjie Zhang, Lijun Chang, and Jianye Yang. 2017. Index-based densest clique percolation community search in networks. *IEEE TKDE transactions on knowledge and data engineering* 30, 5 (2017), 922–935.

[57] Zhiwei Zhang, Xin Huang, Jianliang Xu, Byron Choi, and Zechao Shang. 2019. Keyword-centric community search. In *the 2019 IEEE ICDE international conference on data engineering*. 422–433.

[58] Chen Zhe, Aixin Sun, and Xiaokui Xiao. 2019. Community detection on large complex attribute network. In *Proceedings of the 2019 ACM SIGKDD international conference on knowledge discovery and data mining*. 2041–2049.

[59] Yang Zhou, Hong Cheng, and Jeffrey Xu Yu. 2009. Graph clustering based on structural/attribute similarities. *Proceedings of the VLDB Endowment* 2, 1 (2009), 718–729.

[60] Yuanyuan Zhu, Qian Zhang, Lu Qin, Lijun Chang, and Jeffrey Xu Yu. 2018. Querying cohesive subgraphs by keywords. In *the 2018 IEEE ICDE international conference on data engineering*. 1324–1327.

[61] Zeyuan Allen Zhu, Silvio Lattanzi, and Vahab Mirrokni. 2013. A local algorithm for finding well-connected clusters. In *the 2013 PMLR ICML international conference on machine learning*. 396–404.