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

Anonymous Author(s)

ABSTRACT

In this paper, we study local clustering on 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 indices 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 demonstrate that our proposed algorithm can achieve up to 130% relative effectiveness improvements, while using 10 times less memory than the state-of-the-art algorithm.

1 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 and transactions, insights can be extracted from the graph structure for numerous domains [7, 10]. Nevertheless, many real-world graphs are labeled graphs, which associate the 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 resuts.

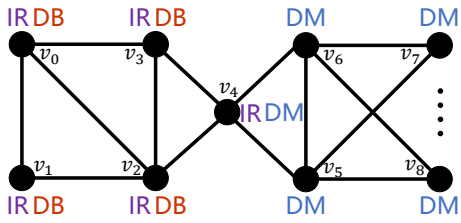


Figure 1: A labeled graph G with the desired cluster $H^* = \{v_0, v_1, v_2, v_3\}$ given the seed node v_0 and query labels {DB, IR}. The dots between v_7 and v_8 represent the remaining nodes in the graph

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 1.1. Figure 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 Figure 1) and areas of interest (e.g., labels DB, 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 checked by further lab experiments [16, 22, 27].

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 [1] [8] and (k, d)-truss[2] [14], which employ topological constraints on the resulting community. Thus, some desirable clusters can be overlooked when they do not satisfy the topological constraints. As verified in our experiments, these methods based on topology-driven models such as ACQ and ATC 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 indices to identify candidate communities satisfying the topological constraints. For large graphs such as `friendster` in our experiments, constructing the state-of-the-art indices [14] 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 more scalable for detecting flexible labeled clusters.

Our work. In this paper, we propose to search local clusters on labeled graphs with a *conductance-driven* approach. Conductance [2, 11, 35], 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 [3, 12, 34, 40] 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 indices 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 each query 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 [1, 2, 17] – 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 pagerank [17] and the inversed pagerank [20].

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 generally NP-hard [31], we propose a two-stage heuristic algorithm for efficient cluster detection. In the first stage, we employ the general sweep algorithm [1, 2, 17] 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 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. 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 query motifs on labeled graphs (Section 3).
- 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 indices due to prohibitive memory consumption. In contrast, our method well supports `friendster` without resorting to costly indices (Section 4).
- 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 0.38 absolute improvement on F1 scores. The two-stage algorithm shows competitive efficiency while using 10x fewer memory than the indexed-based approaches (Section 5).

2 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.

2.1 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

[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.

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 expensive for large graphs [23]. Following existing works [1, 2, 40], 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 [11, 25, 39]. In this work, we follow [14] 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 2.1. Existing works define the structure density as an indicator function of certain topological model. For example, the authors in [14] 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 Figure 1, if we take $k = 1$, $d = 2$ and query labels as $L_q = \{\texttt{DB}, \texttt{IR}\}$, the score of $H^* = \{v_0, v_1, v_2, v_3\}$ is $f_q(H^*) = 1 \cdot \frac{8}{4} = 2$.

Since different variations of $f_q(H)$ have been proposed by existing works, we review them in the remaining part of this section.

2.2 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. Note that there are a plethora of works on local graph clustering or community search for unlabeled graphs [1, 2, 40], which are not the focus of this related work. We refer the interested readers to the survey [15]. The comparsion between these related studies and this paper is summarized in Table 1 and elaborated in the rest of this section.

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

Table 1: A comparison of representative community search works on labeled graphs.

Method	Seed Node	Query Labels	Topological constraints free	Index free
[41, 44]	✘	✔	✘	✘
[5]	✘	✔	✔	✔
[24]	✔	✘	✘	✔
[8, 14]	✔	✔	✘	✘
[9]	✔	✘	✘	✘
Ours	✔	✔	✔	✔

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 [8, 14], 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 [8] and (k, d)-truss [14], and define the structure density in $f_q(H)$ as the indicator function of the model. The effectiveness of these methods heavily depends on users in fine-tuning the hyperparameter of these models. For instance, users need to specify a just-right k value for the k-core model when handling different graphs or even different queries from the same graph [8] since the model only selects subgraphs satisfying the k-core constraints as results. However, in most if not all practical cases when exploring a graph, users have no means to determine such a k. Besides, as discussed in Section 1, these methods requires prohibitively large indices for online processing and thus can't scale to large graphs. 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 [5, 41, 44]. 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, namely node-centric community search (NCS), takes a set of query nodes as the input [24]. The methods in the second category do not support query labels but only find a community where the nodes share similar labels within the community, i.e. the label denstiy function in $f_q(H)$ can not take L_q as parameters.

Recently, meta-path based community search over heterogeneous networks (HCS) was proposed in [9]. 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 [13, 29, 38, 42, 43]. 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

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 Section 4.

3.1 Transformation under LAM

Motivated by the motif-aware graph clustering on unlabeled graphs [3, 12, 34, 40], 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 rather than other more complex motifs because: (1) Triangles in different networks represent stable connections between nodes and are universal building blocks of clusters [30, 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 [28, 32] whereas it is prohibitive 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 to set the edge weights will result in graph fragmentation for unlabeled graphs [21]. 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.

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 3.1. In Figure 1, given the query labels $L_q = \{\mathtt{IR}, \mathtt{DB}\}$ 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 $w_M(v_2, v_3) = 0.5 \times 12 + (1 - 0.5) = 6.5$. Figure 2 shows the LAM graph G_M induced by

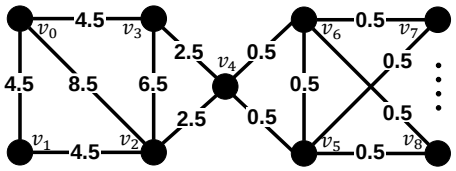


Figure 2: The LAM graph G_M given query labels $L_q = \{\texttt{IR}, \texttt{DB}\}$ and $\lambda = 0.5$.

$L_q = \{\texttt{IR}, \texttt{DB}\}$ 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 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 [17] and the inversed pagerank [20].

The above theoretical guarantee of LAM leads to an easier clustering task. Local graph clustering algorithms [1, 2] 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, [1] 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 personalized pagerank 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 [1, 2, 18], 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 [18] 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 $p_{in} > p_{out}$. 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 shows that LAM concentrates the PPR distribution within H_0 asymptotically.

Theorem 3.2. *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 3.4.

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 3.3. *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 $\bar{d}^j(u)$ 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 [26] 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 $1 - O(N^2e^{-N})$, 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:

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

for any u. By union bounds we further get:

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

which implies Lemma 3.3 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 3.3, we prove the following lemma which directly lead to Theorem 3.2.

Lemma 3.4. *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 3.3, we can prove the Equations 1 and 2 using a similar approach described in the first section of appendix for [18]. 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:

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

$$R^M_t(H_i) \in \big[(1-\epsilon)\bar{R}^M_t(H_i), (1+\epsilon)\bar{R}^M_t(H_i)\big] \quad (2)$$

holds with a probability of at least $1-\delta$ for all $0 < t \leq T$, where $\bar{R}_t(H_i)$ and $\bar{R}^M_t(H_i)$ 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 1 and 2, Lemma 3.4 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:

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

Let $a = \frac{(p^2+pq)}{p^2+2pq+p^2_q} - \frac{(1+pq)}{1+2pq+p^2p^2_q}$, $b = \frac{(1+pq)}{1+2pq+p^2p^2_q}$, 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. □

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 2, when the LAM framework is applied, the PPR value within ground-truth clusters consistently becomes larger across all datasets.

Table 2: 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%

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. 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. 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 $\phi_M(H)$ to measure the 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 [8, 14]. For example, [14] employs $\rho(H, L_q) = \rho_1(H, L_q) = \sum_{l \in L_q} \frac{|V(\{l\}) \cap V(H)|^2}{|V(H)|}$. 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. It is worth highlighting that how to choose the best $\rho(H, L_q)$ is orthogonal to this work and we conduct experiments to show that our general formulation can effectively find labeled clusters for any well-thought $\rho(H, L_q)$.

The following theorem states that optimizing $f_q(H)$ is generally NP-hard.

Theorem 4.1. *It is generally NP-hard to optimizing the cluster metric* $f_q(H) = \frac{\rho(H, L_q)}{\phi_M(H)}$.

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 [33]. □

Discussion. Although there may exist some definitions of $\rho(H, L_q)$ such that optimizing $f_q(H)$ is polynomially solvable, we focus on devising a unified algorithmic framework that can deal with general definitions of $\rho(H, L_q)$. Furthermore, the proof of Theorem 4.1 suggests that approximating a general $f_q(H)$ is at least as hard as approximating the conductance $\phi(H)$. There exist $O(\sqrt{\phi(H^*)})$-approximate algorithms for $\phi(H)$ based on cheeger's inequality [1, 2, 6] where H^* is the cluster with smallest conductance containing the query node. However, these methods cannot be generalized to deal with $f_q(H)$. To this end, we devise a two-stage heuristic algorithm to optimize $f_q(H)$ and leave algorithms with approximation guarantees for future work.

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 $f_q(\cdot)$

```
π_M(s,u) ← 0 for each u ;                                   // Stage I
r(s,u) ← 0 for each u ≠ s and r(s,s) ← 1;
while ∃u : r(s,u) > ε · d(u) do
    π_M(s,u) ← π_M(s,u) + α · r(s,u);
    r(s,u) ← (1 − α) · r(s,u)/2;
    for each v ∈ N(u):
        r(s,v) ← r(s,v) + (1 − α) · r(s,u)·w_M(u,v) / (2·d_M(u));
end
for each u with π_M(s,u) > 0: π_M(s,u) ← π_M(s,u)/d(u);
sort nodes by descending π_M(s) and u_i denotes the i-th
  node after sorting;
H ← arg min_{H_i ∀i≤b} φ_M(H_i), where H_i = {u_1, ..., u_i} ;
ρ ← 0 ;                                                     // Stage II
while ρ < ρ(H, L_q) do
    ρ ← ρ(H, L_q);
    v_r ← arg min_{u∈H} dep(u, H);
    remove v_r from H;
end
H* ← H ∪ {v_r}, where v_r is the last removed node;
return H*
```

4.2 Index-free Two-stage Algorithm

Given that maximizing the cluster metric $f_q(H)$ is NP-hard and NP-hard to approximate, 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 the LAM conductance $\phi_M(H)$. 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 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 [1, 2, 6]. 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 the details of the algorithm as follows.

Stage I: minimizing $\phi_M(H)$ (Lines 1-10). We adopt the sweep algorithm [2] 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 [2], 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 [37] and we set $b = 1000$ constantly 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 according to Theorem 4.1. 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.

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:

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

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 gives the 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 Algorithm 1.

Theorem 4.2. *The time complexity of Algorithm 1 is bounded by $O(\frac{d_{max} \cdot L_{max}}{\epsilon})$.*

Proof. We first give the time complexity of obtaining the approximate PPR distribution over G_M locally. 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 actually, we only have to 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 total time complexity of Algorithm 1 is bounded by the time complexity of Stage I, which is $O(\frac{d_{max} \cdot L_{max}}{\epsilon})$. □

According to Theorem 4.2, 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.

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:

- Can LAM outperform the state-of-the-art solutions for identifying the desired cluster in both real and synthetic datasets (Section 5.2)?
- Is LAM robust to different label distributions (Section 5.3)?
- Does the peeling process indeed boost the results (Section 5.4)?
- Is the index-free 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)?
- Is parameter tuning made easy for the LAM framework (Section 5.7)?

5.1 Experimental Setup

Datasets. Six real-world datasets with ground-truth communities from SNAP [19] are used in our experiments: `facebook`, `amazon`, `dblp`, `youtube`, `livejournal` and `orkut`. The dataset statistics are reported in Table 3.

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.

The other five real-world networks contain ground-truth clusters but do not contain real-world labels. To this end, we follow the setup used in [14] 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.

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.

Table 3: Dataset Statistics. The second-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	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	✘
`sbm`	500	125K	288	0.9	✘

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

The seed node is randomly selected from the cluster to avoid the influence of different seeding algorithms [14]. 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 choose top-3 labels that occuring most frequently in a given cluster and rarely occuring outside the cluster as representative labels [14].

Compared Methods. We compared the proposed methods with the state-of-the-art methods on label-aware community search that are most relevant to our work.

- ATC [14] returns a (k, d)-truss community containing the seed node that maximizes the label score function $\rho_1(H, L_q)$.
- ACQ [8] 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 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 different well-thought $\rho(H, L_q)$.

Table 4: 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

Note that for ACQ and ATC, suitable hyperparameters k and d have to be assigned for each query. We follow the setup in [14] 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 [14].

Parameter Setup. The parameters used in the experiments and their default values are summarized in Table 4. We set the hyperparameter λ of LAM to 0.4 during our experiments. In section 5.7, we show that $\lambda = 0.4$ can consistently achieve the best overall result across all datasets. Compared with the topology-driven community search models which set different k values even for different queries from the same network, tuning the hyperparameter λ is much easier for non-expert users.

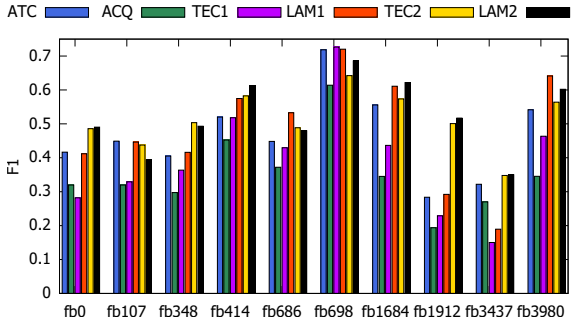
Evaluation Metric. In the experiments, we use the F1-score, which is the harmonic average of precision and recall w.r.t. the ground-truth clusters to validate the *effectiveness* of different algorithms. For evaluating the *efficiency*, we consider the average of time and memory consumption for answering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

Environment. All experiments are conducted on a Linux Server with Intel Xeon Gold 6140 CPU and 256 GB memory running Ubuntu 18.04. We use the original implementation of ATC [14] (implemented with C++) and ACQ [8] (implemented with Java) provided by the authors. Other algorithms are implemented with C++ and executed with a single thread. To guarantee the reproducibility, we will release the source code of LAM in open source.

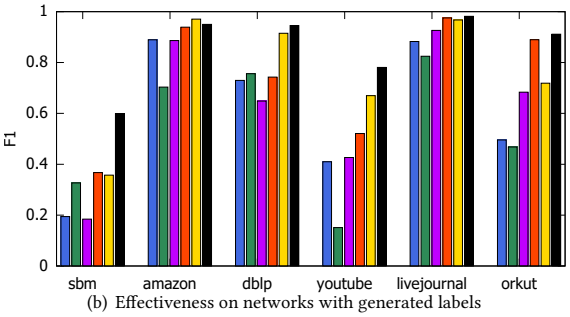
5.2 Effectiveness Evaluation

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 3 reports the F1 scores of the compared methods on each dataset. we make the following observations from the results.

First, the algorithms based on the LAM framework outperform ATC and ACQ on most ego-networks with real-world labels from `facebook` (Figure 3(a)) and all the networks with generated labels (Figure 3(b)). 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 average 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



(a) Effectiveness on `facebook` with real-world labels



(b) Effectiveness on networks with generated labels

Figure 3: Effectiveness comparison.

significantly inferior F1 scores. The impact of ground-truth clusters' structural goodness on effectiveness cannot be distinguished easily over ego networks from `facebook` due to a large number of noises contained in real-world labels. Nevertheless, we can find that on `fb1912` where the ground-truth clusters is illy-structured, LAM2 can achieve 82.34% relative improvements over ATC. 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 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 on 9 out of 10 `facebook` ego-networks and LAM2 outperforms TEC2 on 7 out of 10 `facebook` ego-networks. On networks with generated labels, only 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 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 Figure 4. Overall, LAM1 and LAM2 consistently outperform the baselines with various values of μ. For the baseline methods, when μ is varied from 0.1 to 0.6, all methods except ACQ and TEC1 can achieve satisfactory F1 scores, while ACQ and TEC1 experience 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.

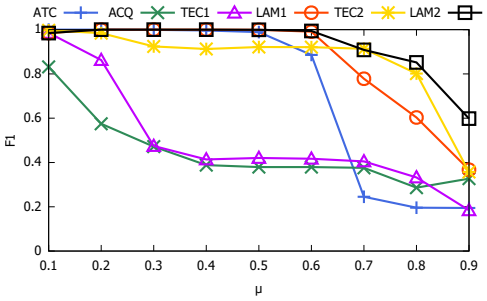


Figure 4: Evaluation on the synthetic networks.

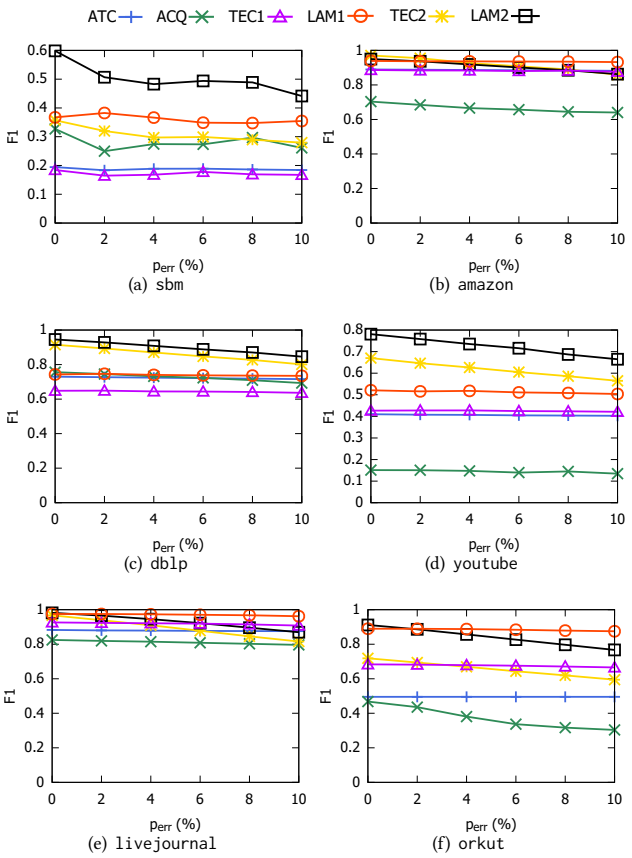


Figure 5: Varying p_{err}.

5.3 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.

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 5 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 methods based on LAM framework (LAM1 and LAM2) still achieves consistently better results than those produced by the topology-driven community search methods (ATC and ACQ) as well as TEC1 and TEC2 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 $\rho_1(H, L_q) = \sum_{l \in L_q} \frac{|V(\{l\}) \cap V(H)|^2}{|V(H)|}$, 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, $\rho_2(H, L_q) = \frac{\sum_u |L_q \cap L(u)|}{|H|}$ 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.

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 6 reports the F1 scores with varying η. As in the previous experiments, our LAM based methods consistently outperform other baselines.

We can see 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.

5.4 Ablation Study of the Peeling Process

In this section, we conduct an ablation study of the peeling process in Stage II. We use LAM$^-$ to represent the algorithm that only optimizes $\phi_M(H)$ through Stage I without conducting the peeling process in Stage II. Similarly, TEC$^-$ represents the corresponding algorithm that only optimizes $\phi(H)$.

Table 5 reports the F1-score of TEC$^-$ and LAM$^-$ compared with corresponding algorithms TEC1 (TEC2) and LAM1 (LAM2). Two key observations can be obtained from this ablation study:

- The peeling process in Stage II is critical and boosts the performance of local clustering on labeled graphs. LAM1 (LAM2) and TEC1 (TEC2) outperform LAM$^-$ and TEC$^-$ significantly across all datasets except on `facebook` where the improvement is relatively small due to a large number of noises contained in real-world labels.
- Our proposed LAM framework is indeed effective for local clustering on labeled graphs. As shown in Table 5, LAM$^-$ outperforms TEC$^-$ consistently at least 12.4% across all datasets and at most 182% on `orkut`.

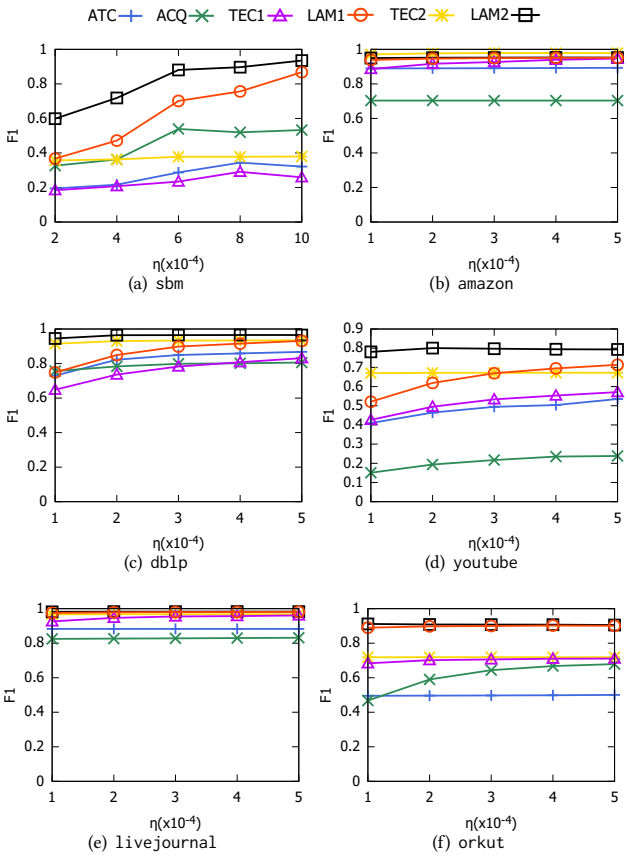


Figure 6: Varying η.

Table 5: Ablation Study of the Peeling Process.

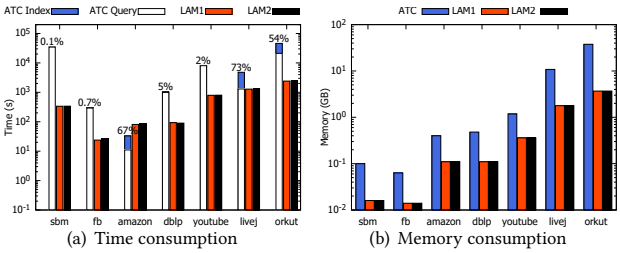
Dataset	TEC$^-$	TEC1/TEC2	LAM$^-$	LAM1/LAM2
sbm	0.15	0.357	0.36	0.598
facebook	0.358	0.393	0.479	0.484
amazon	0.798	0.97	0.897	0.95
dblp	0.473	0.915	0.61	0.945
youtube	0.19	0.67	0.26	0.78
livejournal	0.586	0.97	0.754	0.98
orkut	0.28	0.72	0.79	0.91

5.5 Efficiency and Scalability

In this section, we compare the efficiency and scalability between state-of-the-art topology-driven approach (ATC) and our conductance-driven approaches (LAM1 and LAM2). The purpose is to demonstrate the efficacy of our index-free algorithm. ACQ and TEC1/2 are omitted because they produce inferior cluster qualities than ATC and the corresponding LAM based methods respectively.

Time Consumption. Figure 7(a) compares the running time of our LAM based algorithms and ATC. Note that ATC needs to construct indices before processing queries. Thus, we compare the running time of 1000 queries for LAM based methods with the running time of 1000 queries plus time of index construction for ATC.

We can figure out that on small datasets with well-structured ground-truth clusters such as amazon (with average unweighted



(a) Time consumption (b) Memory consumption

Figure 7: 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.

conductance 0.07), ATC is more efficient than LAM based methods. However, in large datasets or datasets with illy-structured ground-truth clusters, LAM based methods are more efficient than ATC. For example, on the large livejournal dataset with well-structured clusters, although the query processing of ATC based on indices can be as efficient as our LAM based methods, index construction dominates the time consumption (73%) and makes ATC as a whole much less efficient than LAM based methods. On the datasets such as sbm, youtube and orkut, LAM based methods process queries more efficient than ATC. In summary, our proposed methods are more efficient than ATC in terms of time consumption.

Memory Consumption. Figure 7(b) reports the memory consumption of LAM based methods and ATC. Due to large indices required by ATC, it consumes significantly more memory than index-free LAM based methods. The gap on memory consumption between these two methods scales w.r.t. the size of the network. For small networks facebook, dblp, youtube, ATC requires 3 to 4 times memory than LAM based methods. For larger networks livejournal, orkut, ATC requires up to 10 times memory than LAM based methods. This implies that our methods are much more efficient than ATC in terms of memory consumption.

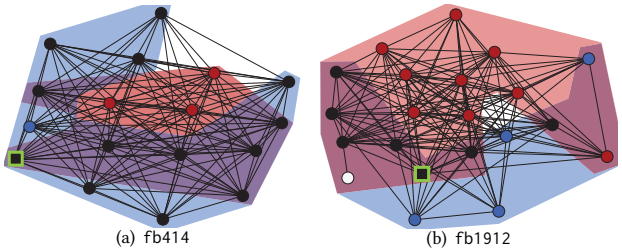
We can also observe that the time and memory consumption of LAM1 and LAM2 is almost the same with each other. This means that the efficiency of our method is not influenced by different choice of $\rho(H, L_q)$.

Scalability. We further compare the scalability of LAM1 and LAM2 with ATC using an extremely large network friendster from SNAP [19] 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 50 seconds on average to process one query for LAM1 and LAM2. Furthermore, the performance can be easily accelerated by parallel processing [4]. Thereby, our methods based on LAM is scalable to handle large networks.

5.6 A Case Study on facebook

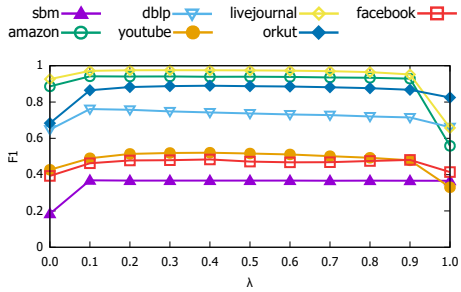
In this subsection, we further validate that LAM-based methods can extract label-aware and cohesive clusters through a case study on facebook dataset.

In Figure 8, we visualize two clusters extracted by different query labels l_1 and l_2 from ego-networks fb414 and fb1912. The query



(a) fb414 (b) fb1912

Figure 8: 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.



(a) Varying λ for LAM1

(b) Varying λ for LAM2

Figure 9: Parameter Tuning.

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 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.

5.7 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 9.

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 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 3), 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 9, 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.

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 suffer from the requirement of strict topology constraints and prohibitively large indices. 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 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] Reid Andersen and Fan Chung. 2007. Detecting sharp drops in PageRank and a simplified local partitioning algorithm. In *TAMC*. 1–12.
[2] Reid Andersen, Fan Chung, and Kevin Lang. 2006. Local graph partitioning using pagerank vectors. In *FOCS*. 475–486.

[3] 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.

[4] Bahman Bahmani, Kaushik Chakrabarti, and Dong Xin. 2011. Fast personalized pagerank on mapreduce. In *SIGMOD*. 973–984.

[5] Lu Chen, Chengfei Liu, Kewen Liao, Jianxin Li, and Rui Zhou. 2019. Contextual community search over large social networks. In *ICDE*. 88–99.

[6] Fan Chung. 2007. The heat kernel as the pagerank of a graph. *Proceedings of the National Academy of Sciences* 104, 50 (2007), 19735–19740.

[7] Elizabeth M Daly and Mads Haahr. 2007. Social network analysis for routing in disconnected delay-tolerant manets. In *MOBIHOC*. 32–40.

[8] 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.

[9] Yixiang Fang, Yixing Yang, Wenjie Zhang, Xuemin Lin, and Xin Cao. 2020. Effective and efficient community search over large heterogeneous information networks. *Proceedings of the VLDB Endowment* 13, 6 (2020), 854–867.

[10] 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.

[11] David F Gleich and C Seshadhri. 2012. Vertex neighborhoods, low conductance cuts, and good seeds for local community methods. In *SIGKDD*. 597–605.

[12] Ling Huang, Chang-Dong Wang, and Hong-Yang Chao. 2018. A harmonic motif modularity approach for multi-layer network community detection. In *ICDM*. 1043–1048.

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

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

[15] Xin Huang, Laks VS Lakshmanan, and Jianliang Xu. 2017. Community search over big graphs: Models, algorithms, and opportunities. In *ICDE*. 1451–1454.

[16] Edward L Huttlin, Raphael J Bruckner, Joao A Paulo, Joe R Cannon, Lily Ting, Kurt Baltier, Greg Colby, Fana Gebreab, Melanie P Gygi, Hannah Parzen, et al. 2017. Architecture of the human interactome defines protein communities and disease networks. *Nature* 545, 7655 (2017), 505–509.

[17] Kyle Kloster and David F Gleich. 2014. Heat kernel based community detection. In *SIGKDD*. 1386–1395.

[18] 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.

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

[20] Pan Li, I Chien, and Olgica Milenkovic. 2019. Optimizing generalized pagerank methods for seed-expansion community detection. In *NeurIPS*. 11710–11721.

[21] Pei-Zhen Li, Ling Huang, Chang-Dong Wang, and Jian-Huang Lai. 2019. EdMot: An edge enhancement approach for motif-aware community detection. In *SIGKDD*. 479–487.

[22] Xiaoli Li, Min Wu, Chee-Keong Kwoh, and See-Kiong Ng. 2010. Computational approaches for detecting protein complexes from protein interaction networks: a survey. *BMC genomics* 11, 1 (2010), 1–19.

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

[24] Qing Liu, Yifan Zhu, Minjun Zhao, Xin Huang, Jianliang Xu, and Yunjun Gao. 2020. VAC: Vertex-Centric Attributed Community Search. In *ICDE*. 937–948.

[25] Farnaz Moradi, Tomas Olovsson, and Philippas Tsigas. 2014. A local seed selection algorithm for overlapping community detection. In *ASONAM*. 1–8.

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

[27] Gergely Palla, Imre Derényi, Illés Farkas, and Tamás Vicsek. 2005. Uncovering the overlapping community structure of complex networks in nature and society. *nature* 435, 7043 (2005), 814–818.

[28] Roger Pearce. 2017. Triangle counting for scale-free graphs at scale in distributed memory. In *HPEC*. 1–4.

[29] Yiye Ruan, David Fuhry, and Srinivasan Parthasarathy. 2013. Efficient community detection in large networks using content and links. In *WWW*. 1089–1098.

[30] 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.

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

[32] Julian Shun and Kanat Tangwongsan. 2015. Multicore triangle computations without tuning. In *ICDE*. 149–160.

[33] Jiří Šíma and Satu Elisa Schaeffer. 2006. On the NP-completeness of some graph cluster measures. In *SOFSEM*. 530–537.

[34] Charalampos E Tsourakakis, Jakub Pachocki, and Michael Mitzenmacher. 2017. Scalable motif-aware graph clustering. In *WWW*. 1451–1460.

[35] 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.

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

[37] 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.

[38] Jaewon Yang, Julian McAuley, and Jure Leskovec. 2013. Community detection in networks with node attributes. In *ICDM*. 1151–1156.

[39] Hao Yin, Austin R Benson, and Jure Leskovec. 2019. The local closure coefficient: A new perspective on network clustering. In *WSDM*. 303–311.

[40] Hao Yin, Austin R Benson, Jure Leskovec, and David F Gleich. 2017. Local higher-order graph clustering. In *SIGKDD*. 555–564.

[41] Zhiwei Zhang, Xin Huang, Jianliang Xu, Byron Choi, and Zechao Shang. 2019. Keyword-centric community search. In *ICDE*. 422–433.

[42] Chen Zhe, Aixin Sun, and Xiaokui Xiao. 2019. Community detection on large complex attribute network. In *SIGKDD*. 2041–2049.

[43] 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.

[44] Yuanyuan Zhu, Qian Zhang, Lu Qin, Lijun Chang, and Jeffrey Xu Yu. 2018. Querying cohesive subgraphs by keywords. In *ICDE*. 1324–1327.