Uncovering Local Hierarchical Overlapping Communities at Scale

Panagiotis Liakos, Alexandros Ntoulas, and Alex Delis

Abstract—Real-life systems involving interacting objects are typically modeled as graphs and can often grow very large in size. Revealing the community structure of such systems is crucial in helping us better understand their complex nature. However, the ever-increasing size of real-world graphs and our evolving perception of what a community is, make the task of community detection very challenging. A critical relevant challenge is the discovery of the possibly overlapping communities of a given node in a billion-node graph. This problem is very common in modern large social networks like Facebook and LinkedIn. In this work, we propose a scalable local community detection approach to efficiently unfold the communities of individual target nodes in a given network. Our goal is to reveal the clusters formed around nodes (e.g., users) by leveraging the relations within all *different* contexts these nodes participate in. Our approach, termed Local Dispersion-aware Link Communities or LDLC, considers the similarity of pairs of links in the graph as well as the extent of their participation in multiple contexts. Then, we determine the order in which we should group the pairs of links so that we form meaningful hierarchical communities. We are not affected by constraints existing in previous techniques such as the need for several seed nodes or the need to collapse multiple overlapping communities to a single community. Our experimental evaluation using ground-truth communities for a wide range of large real-world networks shows that our LDLC algorithm significantly outperforms state-of-the-art methods on both accuracy and efficiency. Moreover, we show that LDLC uncovers very effectively the hierarchical structure of overlapping communities by producing detailed dendrograms.

Index Terms—Community detection, complex networks, hierarchical communities, dispersion.

1 Introduction

The neurons in our brains, the proteins in live cells, the powerplants of an electrical grid, and the users of an online social networking service, are all entities of *complex systems* that play a vital role in our daily lives. Networks are a powerful tool for modeling relations and interactions between the components of such complex systems. Respective real-world networks are often massive; yet they exhibit a high level of order and organization, which allows the study of common properties they exhibit, such as the power-law degree distribution and the small-world structure [1], [2]. Another important property that real-world networks exhibit is the presence of community structure [3]. At a high level, communities are groups of nodes that share a common functional property or context, e.g., two people that attended the same school, or two movies with the same actor. In several cases communities in a network are distinct; consider for example the fans of different basketball teams. However, it is often the case that communities overlap. Figure 1 illustrates the communities of an individual in a social network, i.e., her family, co-workers, basketball buddies and friends from college. It is obvious that the communities may overlap in different ways. For example, a co-worker may also be a basketball buddy and a friend from college. Such overlapping communities may have a complex structure of connections that are not easy to discern and are certainly more challenging to identify compared to non-overlapping ones.

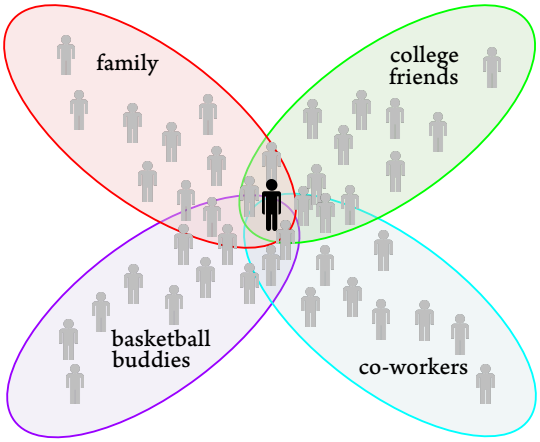

Fig. 1. Illustration of the social circles of an individual. Her family, co-workers, basketball buddies and friends from college are distinct yet overlapping communities.

Effectively extracting the community structure of a node in a network has many useful applications:

- We can provide more informative and engaging social network feeds by better understanding the membership of an individual to various organizational groups.
- We can suggest common friends of an individual to connect because they share mutual interests.
- We can create match-making algorithms for online players based on the similarity of their game play.
- We can identify groups of customers with similar behavior and enhance the efficiency of recommender sys-

- *P. Liakos is with the University of Athens, Athens 15703, Greece. E-mail: p.liakos@di.uoa.gr*
- *A. Ntoulas is with the University of Athens, Athens 15703, Greece. Email: antoulas@di.uoa.gr*
- *A. Delis is with the University of Athens, Athens 15703, Greece, and the New York University Abu Dhabi, United Arab Emirates. Email: ad@di.uoa.gr*

Manuscript received March 18, 2018; revised 07/05/2019.

tems.

Early community detection approaches focused either on grouping the nodes of a network or on searching for links that should be removed to separate the clusters [4]. However, these approaches did not consider the fact that communities may overlap, and ultimately could not provide an accurate representation of a network's community structure. Algorithms that followed [5], [6], [7], [8], [9], [10] allow for nodes to belong to several overlapping communities by employing techniques such as link clustering, matrix factorization, and personalized PageRank vectors. Still, these approaches are not applicable to the massive graphs of the *Big Data* era, as they focus on the *entire* graph structure and do not scale with regards to both execution time and memory consumption. Recent efforts have therefore shifted the focus from the global structure to a local view of the network [11], [12], [13], [14]. More specifically, such approaches locally expand a set of target nodes in the community of interest, instead of uncovering the communities of the entire network.

Seed set expansion approaches employ techniques such as random walks to estimate the likelihood of a node to participate in the target community, and manage to scale to large networks [11], [12], [13], [14]. These approaches consider that overlaps between communities are sparsely connected whereas the areas where communities overlap are denser than the actual communities. However, studies of real-world networks show that two nodes are more likely to be connected if they share multiple communities in common [15]. For example, people belonging to both the co-workers and basketball buddies communities of Figure 1, are expected to know each other with high probability. Hence, as the overlapping area is in fact denser than the actual communities, seed set expansion approaches are driven towards nodes that reside in the overlap. In addition to this, all scalable methods require *multiple seeds* to avoid detecting multiple overlapping communities as a single one. This constitutes a challenge, as it is usually the case that we are interested in all communities of a single node, instead of seeking one community involving multiple predefined nodes. Finally, seed set expansion approaches are shown to perform well when detecting relatively large communities, whereas high quality communities are in fact small [15].

In this paper, we focus on the neighbors of a single node in the network, i.e., its egonet, and aim at extracting the –possibly overlapping– communities of this node. We build upon the ideas of *link clustering* [5], [6] and employ *similarity* measures that allow for effectively handling densely connected overlaps between communities. Our intuition is that when grouping pairs of links we should capture the *extent* to which a link belongs to multiple overlapping communities. To this end, we utilize a dispersion-based tie-strength measure that helps us quantify the participation of a link's adjacent nodes to more than one community. Our approach is both *efficient* and *scalable* as we focus on local parts of graphs comprising a target node and its neighbors. As we show in our experimental evaluation, we produce a more accurate and intuitive representation of the community structure around a node for a number of real-world networks.

In summary, we make the following contributions:

- We propose a local community detection algorithm that effectively reveals the overlapping nature of real-world network communities of individual target nodes.
- We operate with less input from the user (a single seed vs a set of multiple seeds) and generate communities of equal or better quality.
- We experimentally evaluate our algorithm against state-of-the-art approaches using publicly available networks. Our results show that our approach significantly outperforms current methods using popular evaluation metrics.
- We reduce the execution time notably, by focusing on the neighborhood of a node and thus, manage to handle billion-edge scale graphs.
- We provide a detailed view of the rich hierarchical structure of the derived community.

Our paper is organized as follows: We first introduce definitions and metrics that will be used in describing our approach in Section 2. In Section 3, we describe our local hierarchical overlapping community detection algorithm named *Local Dispersion-aware Link Communities* (`LDLC`), and explain its rationale. In Section 4, we extensively evaluate our approach both qualitatively and quantitatively. Section 5 reviews related work and finally, Section 6 concludes our paper.

2 Background

In this section we review some basic principles and definitions for our work. First, we provide the definition of the *egonet* and subsequently we discuss measures that are used to estimate the strength of *ties* in networks. Finally, we give the definition of *partition density* and detail the dataset used in our study.

2.1 Egonet

Large-scale graph mining applications are often based on local neighborhoods of nodes [16]. The set of nodes that are one hop away from a given node allows for a variety of analyses that build intuition about that node and its role. Focusing on local neighborhoods of nodes enables respective applications to scale effortlessly to large networks as the task in hand can be executed in parallel for all nodes in the network. In the context of social networks, this one hop neighborhood is frequently called the *egonet* of a user. Figure 1 depicts such an egonet of an individual and the overlapping communities she is part of.

In this work, we address the challenge of extracting efficiently the community structure formed by the nodes adjacent to a *single* target node. The networks we are interested are often massive and, thus, our approach should scale to graphs of extreme volume. To this end, we investigate ground-truth communities of real-world networks and in particular, whether these communities are recoverable using egonets alone. Eventually, we focus on the egonets of target nodes to significantly reduce the search space of our algorithm.

2.2 Tie Strength Measures

The *closeness* between nodes in a network and its impact on the network's dynamics has been studied extensively [17], [18]. Understanding the complex nature of interacting objects calls for quantifying the *strength* of ties to distinguish the connections of particular importance. We outline here the tie strength measures that we employ in the context of this work:

2.2.1 Embeddedness

Intuitively, a large number of shared neighbors between nodes indicates a *strong* tie, whereas a few mutual neighbors indicate a *weak* tie. Therefore, a frequently used measure to estimate the tie strength between two nodes u and v is *embeddedness*, formally defined as:

$$emb(u, v) = |N_+(u) \cap N_+(v)| \tag{1}$$

where $N_+(u)$ is the set of nodes adjacent to u.

In the case of social networks, individuals operating on a common context are more likely to share joint activities with each other, as opposed to people that do not share this particular context [19]. Therefore, *embeddedness* can effectively be applied for the identification of couples [20].

2.2.2 Jaccard similarity coefficient

The Jaccard similarity coefficient is a frequently used measure of similarity of two sets and is defined as the size of their intersection divided by the size of their union. In the case of two nodes u and v in a network, the Jaccard similarity coefficient can be applied on the respective sets of neighbors, $N_+(u)$ and $N_+(v)$, as follows:

$$J(u, v) = \frac{|N_+(u) \cap N_+(v)|}{|N_+(u) \cup N_+(v)|} \tag{2}$$

2.2.3 Absolute and Recursive Dispersion

The task of identifying spouses or romantic partners in a social network is also the focus of [21]. Backstrom and Kleinberg address this challenge through the use of *dispersion*-based measures. They analyze real data from Facebook and conclude that high dispersion is indeed present, not only to spouses or romantic partners, but to people who share multiple relevant aspects of their social environment in general.

Formally, we consider the egonet G_u of u in G and define *absolute dispersion* as:

$$disp(u, v) = \sum_{\substack{s,t \in C_{uv} \\ s<t}} d_v(s, t) \tag{3}$$

where C_{uv} is the set of common neighbors of u and v in G_u, and $d_v(s, t)$ is a distance function equal to 1 when s and t are not directly linked themselves and have no common neighbors in G_u other than u and v, and 0 otherwise.

Experiments show that for a fixed value of $disp(u, v)$, increased embeddedness is a negative predictor of whether v is *close* to u [21]. Thus, absolute dispersion is more effective when normalized using embeddedness. In addition, the performance of dispersion is found to strengthen when applying it recursively as follows. First, we consider $x_v = 1$ for all neighbors v of u. Then, we iteratively update x_v using the formula:

$$x_v = \frac{\sum_{w \in C_{ij}} x_w^2 + 2 \sum_{\substack{s,t \in C_{ij} \\ s<t}} d_v(s, t) x_s x_t}{emb(u, v)} \tag{4}$$

The value produced after the *third iteration* of (4) is empirically found to perform the best [21]. We will refer to this value as *recursive dispersion* of v in G_u for the rest of this paper, and use it to identify pairs of nodes that operate in multiple common contexts.

2.3 Partition Density

Agglomerative community detection algorithms provide us with a dendrogram describing the hierarchical organization pattern of communities [4], [5], [22]. To obtain meaningful communities from the dendrogram it is necessary to determine the level at which to cut the tree at. To this end, Ahn et al. [5] introduced the measure of *partition density* D, that is formally defined as follows:

$$D = \frac{2}{|E|} \sum_{c \in C} e_c \frac{e_c - (n_c - 1)}{(n_c - 2)(n_c - 1)} \tag{5}$$

where C is the set of communities discovered, e_c is the number of links in a community $c \in C$, and n_c is the number of nodes all the links in e_c touch.

We can come up with the optimal value of D by examining its value at each step of the hierarchical clustering process. Cutting the dendrogram at the level that produces the optimal value of D is shown to effectively derive meaningful and relevant communities. In addition, partition density is suitable for large-scale graphs as it does not suffer a resolution limit like *modularity* [23], being that every term in Equation (5) is local in each community c.

2.4 Networks in our Dataset

Evaluating and comparing communities detected by different algorithms is not a trivial task. Large networks exhibit extremely complex organization and cannot be visualized in meaningful ways. However, we can measure the accuracy of a community detection algorithm given the presence of *ground-truth* communities [15].

In this work, we employ *all six* of the real-world networks with available ground-truth communities that are provided by the Stanford Network Analysis Project (SNAP).[1] In particular, our evaluation is based on the top-$5,000$ highest quality communities of each of these networks [24]. Table 1 provides the details of our dataset.

DBLP is a co-authorship network and ground-truth is formed from authors who published in the same journal or conference. *Amazon* is a product co-purchasing network and the annotated communities associated with it are based on the categories of the products. Finally, *Youtube*, *LiveJournal*, *Orkut*, and *Friendster* are all social networks, and the respective ground-truth communities are user groups that have been formed in these networks. We note that Table 1 features a graph that exceeds 1.8 billion edges, namely *Friendster*. We

[1] https://snap.stanford.edu/data/#communities

TABLE 1
Graphs of our dataset reaching up to 1.8 billion edges.

Graphs	Type	Nodes	Edges	Av. Degree	Av. Community Size
DBLP	Co-authorship	317,080	1,049,866	3.31	22.45
Amazon	Co-purchasing	334,863	925,872	2.76	13.49
Youtube	Social	1,134,890	2,987,624	2.63	14.59
LiveJournal	Social	3,997,962	34,681,189	8.67	27.80
Orkut	Social	3,072,441	117,185,083	38.14	215.72
Friendster	Social	65,608,366	1,806,067,135	27.53	46.81

also see that, the average community size of most networks is relatively small, with the exception of *Orkut* with an average size of 215.72.

3 Local Dispersion-aware Link Communities

In this section we describe in detail our approach for local community detection. We commence by examining the coverage ratio of egonets on the ground-truth communities of the networks in our dataset. We then discuss the difficulties that existing methods based on seed set expansion and link clustering face due to the nature of real-world overlapping communities. We show that the use of dispersion-based measures of tie strength can alleviate such issues. Then, we present our algorithm, termed `LDLC`, in detail. Finally, we discuss a novel sampling technique to effectively reduce the search space of our algorithm.

3.1 Egonet Coverage Ratio

Community detection methods that focus on the *global structure* of graphs fail to scale to the massive volume that real-world networks reach, i.e., millions of nodes and billions of edges. We aim at detecting communities for large-scale graphs efficiently. To this end, we focus on the *local* structure of a node in the network. Studies of real-world networks show that community members tend to organize themselves around hub nodes that are linked with most of the nodes in the community [15]. We begin discussing our approach by investigating ground-truth communities of real-world networks, and in particular, the fraction of the nodes they comprise that is part of *egonets* of nodes that belong to the respective communities.

We report in Figure 2 the coverage ratio of egonets regarding the ground-truth communities of the networks of our dataset. For every ground-truth community of all six networks of our dataset, we examined the coverage of the egonets of each of the nodes belonging to the community. The average coverage ratio depicted in Figure 2, results from the egonets of the nodes with the largest coverage for each ground-truth community. We observe that the coverage ratio is very high for all networks, ranging from 87% to 97%, with the exception of *Orkut* at slightly under 67%. The lower coverage ratio of *Orkut* is attributed to the larger average community size of this network. Empirical observations [15] show that high quality communities usually consist of no more than 100 nodes, whereas the average community size of *Orkut* is more than twice as high and remains low even when using the 2-step geodesic neighborhood of nodes, as reported in [11].

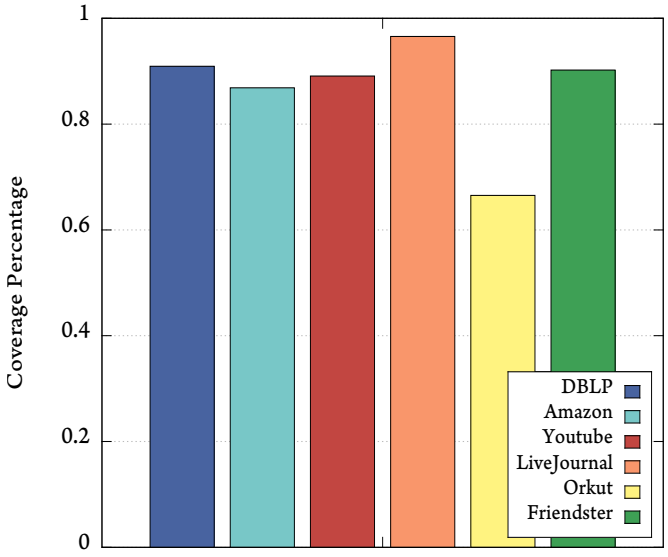

Fig. 2. Egonet coverage ratio for the ground-truth communities of the different graphs provided by SNAP. We show that the coverage ratio for all graphs, with the exception of *Orkut*, is above 87%. The ratio is lower for *Orkut* due to its large average ground-truth community size.

The large coverage ratio of egonets on ground-truth communities verifies our hypothesis that high quality communities can be detected when focusing on egonets of nodes. This allows us to significantly reduce the scale of our search by focusing only on a small part of a possibly extremely large network. Even in the case of nodes exhibiting large degrees, dealing with the respective egonets instead of the global structure of the graph is beyond comparison with regard to efficiency. Space complexity is also reduced greatly, as the memory footprint of the egonet is insignificant when compared to the whole network.

3.2 Effective Detection of Local Hierarchical Overlapping Communities

Investigations on the structure of real-world networks have revealed that there is an increasing relationship between the number of shared communities and the probability of nodes being linked with an edge [15]. Hence, the nodes residing in overlapping parts of communities are more densely connected than the nodes residing in the non-overlapping parts.

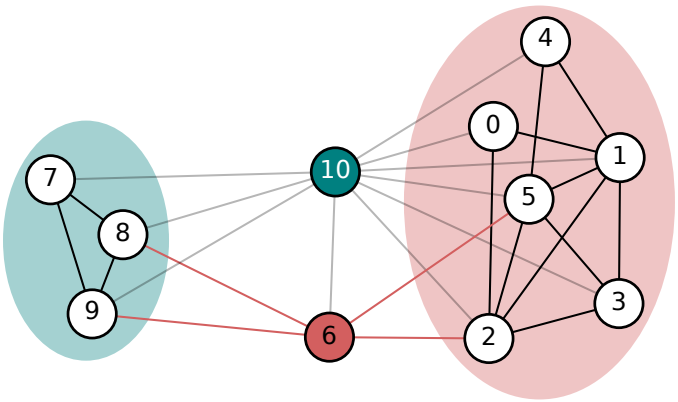

Fig. 3. Social communities in the egonet of an individual (**10**) in a social network. Using a force-directed layout we can easily identify two well-connected groups of acquaintances. A special tie between (**10**) and (**6**) is evident, as (**6**) is the only vertex having links (red colored) towards both communities.

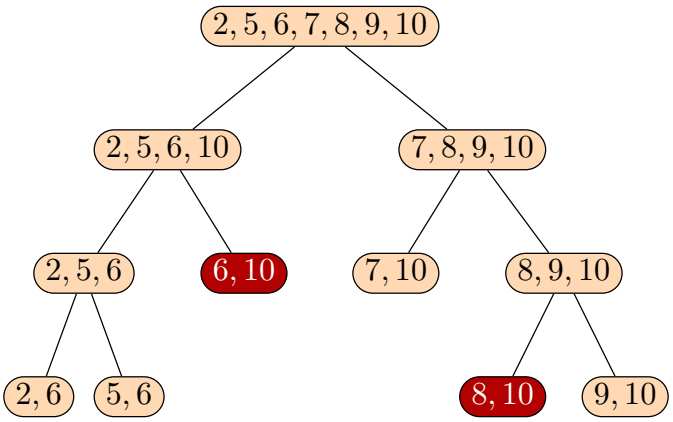

Fig. 4. The hierarchical link structure of the malformed community that results when performing link clustering in the egonet of Figure 3 using Equation (6), and cutting at the level of optimal partition density. The similarity of link (6,10) with link (8,10) leads to a community that groups numerous nodes that are not linked with each other.

Moreover, *connector* nodes, i.e., nodes linked with most of the members of a community, belong to the overlap [15].

Recent local community detection methods [11], [12], [14] expand seed sets by utilizing the dynamics of random walks initiating from the seeds. The participation of a node in the target local community is determined by the corresponding probability score that results from these random walks. Naturally, nodes that reside in the dense overlapping area of multiple communities of a particular node, have high probability scores for random walks starting off this node. In addition to this, nodes outside the overlapping area that are selected in the resulting community due to their probability scores, do not necessarily belong to the same community, as each random walk starting from a seed node is likely to reach a different community. Hierarchical link clustering approaches focusing on the global network structure [5], [6] examine the similarity of pairs of links, and thus, avoid grouping nodes that actually belong to disparate communities. However, such approaches do consider that communities in whole are more densely connected than their overlapping parts [25]. Therefore, these approaches are also unable to handle overlaps appropriately.

Figure 3 illustrates the egonet of an individual (**10**) in a social network. We use this egonet here as a *toy example*. The use of a force-directed layout enables us to easily identify the organizational groups shaped around this node. In particular, we observe that the neighbors of node **10** form two well-connected groups. We also notice, that the only node in the egonet that maintains links (red-colored) with nodes of both groups other than **10**, is **6**. The relationship between nodes **10** and **6** is a particular case of a strong tie which is frequent in networks and has to be considered when identifying overlapping communities. Node **6** acts as a *connector* in the egonet of **10** and is linked with nodes that are not themselves well-connected, as they belong to different organizational groups.

3.2.1 Local hierarchical link communities

We address the task of local community detection by merging pairs of links in the egonet of a target node. Links often demonstrate a particular relation, e.g., a friendship between two nodes, whereas nodes are usually part of multiple groups. Thus, by grouping links –instead of nodes– we allow for the participation of nodes into multiple overlapping communities. To quantify the relevance of two edges e_{wu} and e_{wv} sharing a common node w, we can properly adopt the Jaccard similarity coefficient in the context of links [5]. Using the common node of the two links provides no additional information, and may introduce bias, depending on the degree of w. Therefore, using Equation (2), we can define the similarity of the pair (e_{wu}, e_{wv}) as:

$$J(e_{wu}, e_{wv}) = J(u, v) = \frac{|N_+(u) \cap N_+(v)|}{|N_+(u) \cup N_+(v)|} \qquad (6)$$

where u and v are both adjacent to w.

After quantifying the similarity of all pairs of links in the egonet of Figure 3 that share a common node using Equation (6), we can build a hierarchy of communities through agglomerative clustering. More specifically, we proceed by merging pairs of links by descending order of similarity. Finally, we cut the resulting dendrogram at the level of optimal partition density, and the communities we come up with are:

1) {**0, 1, 2, 3, 4, 5, 10**}
2) {**6, 7, 8, 9, 10**}
3) {**2, 5, 6, 7, 8, 9, 10**}

We see that the first two communities are *well-knit*, i.e., the respective sub-graph is quite dense. However, we also observe that the third community coalesces numerous nodes that are not linked together (**2, 5** with **7, 8, 9**). This is an effect of the Jaccard similarity coefficient that is used for quantifying the similarity of pairs of links, as this measure is unable to capture *how well* the neighbors of two nodes are interconnected. Indeed, the hierarchical link structure of this particular community, as portrayed through Figure 4, highlights evidently the cause of this undesired behavior. We see that the third community actually results from the coalescence of two well-separated clusters (**2, 5, 6, 10** and **7, 8, 9, 10**). This grouping occurs due to the similarity of links **(6, 10)** and **(8, 10)**. These links are in fact similar when considering the Jaccard similarity coefficient; however they

belong to an overlapping area of different communities that Equation (6) is unable to take into account and consequently reveal. In addition, as the distance between two clusters is determined by a single pair, grouping large clusters with each other results in a dendrogram that reveals very little about the hierarchical structure of the community.

3.2.2 Building on dispersion-based measures

Estimating the relevance of pairs of links in the presence of dense community overlaps calls for measures that take into account the extent to which the neighbors of two nodes are interconnected. Dispersion-based measures address this challenge and therefore fit perfectly in the task of overlapping community detection. Through their use, we are able to *single out connector nodes* that lie in overlapping parts of communities. For example, using Equation (3) we obtain that node **6** exhibits the highest absolute dispersion in the egonet of **10** with a value of 4. Hence, we can employ dispersion-based measures to favor groupings of pairs of links with adjacent nodes that share a lot of common neighbors (high Jaccard similarity coefficient) only if these neighbors are also well interconnected (low recursive dispersion). In this way, connector nodes are involved in groupings at a *higher* level of the resulting dendrogram, which then depicts more accurately the hierarchical structure of the communities in the egonet. In particular, we propose the use of the recursive dispersion measure *along with* the Jaccard similarity coefficient in order estimate the relevance of pairs of links. More formally, we define the similarity S of two pairs of links (e_{wu}, e_{wv}) to be:

$$S(e_{wu}, e_{wv}) = \frac{J(e_{wu}, e_{wv})}{rec(u) + rec(v) + rec(w)} \tag{7}$$

where $rec(u)$ is the recursive dispersion of u in the egonet of the target node.

Returning on the example of Figure 3 and the egonet of node 10, if we apply hierarchical link clustering using Equation (7) as a measure of similarity instead of (6), we come up with the following communities:

1) {**0, 1, 2, 3, 4, 5, 10**}
2) {**7, 8, 9, 10**}
3) {**2, 5, 6, 10**}
4) {**6, 8, 9, 10**}

We observe that the nodes of all communities are much more well-connected. Moreover, node **6** is featured in two communities, in which every two distinct vertices are adjacent, i.e., they form *cliques*. Evidently, through Equation (7), we are able to penalize the high dispersion that node **6** exhibits in this egonet. Links featuring this node are now merged at a higher level of the resulting dendrogram, and thus, we avoid forming communities featuring nodes of different organizational groups.

3.3 Our Proposed LDLC Algorithm

We present here the LDLC algorithm for finding local hierarchical overlapping communities in large-scale graphs. LDLC is an agglomerative clustering algorithm whose aim is to reveal the hierarchical structure of the possibly overlapping communities of a single target node in a network. LDLC uses Equation (7) to determine the similarity of all pairs of links in the egonet of the target node in the network that share a common node. These pairs of links are merged progressively in ranking order according to their similarity. The groupings result in a dendrogram that depicts the hierarchical organization of the communities the target node belongs to. To derive the actual communities, we may cut this dendrogram at the level that produces the optimal partition density (Equation (5)), or alternatively, we can cut it at the level that produces the desired number of communities.

Algorithm 1: LDLC(G, u)

input : An undirected network $G = (V, E)$, and a node $u \in V$.
output : A dendrogram depicting the hierarchical (possibly overlapping) communities of G_u.

```
1  begin
2      G_u(V_u, E_u) ← egonet(G, u);
3      rec ← dict();
4      foreach v ∈ G_u, v ≠ u do
5          rec[v] ← 1;
6      for iteration ← 1 to 3 do
7          foreach v ∈ V_u, v ≠ u do
8              rec[v] ← (Σ_{w∈C_uv} rec[w]^2 + 2 Σ_{s,t∈C_uv, s<t} d(s,t) rec[s] rec[t]) / emb(u,v);
9      similarities ← min_heap();
10     for k ∈ V_u do
11         for (e_ik, e_jk) ← combinations(N_+(k), 2) do
12             J(e_ik, e_jk) ← |N_+(i) ∩ N_+(j)| / |N_+(i) ∪ N_+(j)|;
13             S(e_ik, e_jk) ← J(e_ik, e_jk) / (rec[i] + rec[j] + rec[k]);
14             similarities ← (1 − S(e_ik, e_jk), (e_ik, e_jk));
15     foreach (similarity, (e_ij, e_jk)) ∈ similarities do
16         join_clusters(e_ik, e_jk);
17         if len(clusters) == 1 then
18             break;
```

Algorithm 1 outlines our suggested LDLC. The input of our algorithm comprises an undirected graph $G(V, E)$ and a single target node $u \in V$. The output of LDLC is a dendrogram depicting the rich hierarchical structure of the local communities of node u.

Loading the egonet and initializing the communities: We start by loading in memory the egonet of u, i.e., node u, its adjacent nodes, and the edges among them (Line 2). Depending on the network representation, this process can be quite costly. For example, using an edge-list or an adjacency-list stored in a file would necessitate two passes over the files, to identify the neighbors of u as well as the neighbors of u's neighbors. To alleviate this cost we focus instead on space-efficient in-memory representations [26], [27], [28], [29], [30]. Such approaches allow for fast neighbor queries as they maintain adjacency lists in-memory, usually in sorted order. Thus, to come up with the egonet of u, we first retrieve the adjacency list of u, and then the adjacency lists of u's neighbors. For each of the latter adjacency lists we keep only those nodes that are neighbors of u, by applying intersection with the adjacency list of u. This operation ends up with u's egonet as it discards all nodes that are not

adjacent to u, as well as the respective edges. Having loaded the egonet, we proceed by initializing the communities. More specifically, we consider every edge $e \in E_u$ to be a community of its own, with the two adjacent nodes as its members.

Computing the recursive dispersion of u's neighbors: Lines 3–8 compute the recursive dispersion of all neighbors of u, $v \in V_u$. We map the respective recursive dispersion values of the neighbors in a dictionary (associative array) that uses the nodes as keys (Line 3). We first assign a value of 1 for all nodes (Lines $4-5$) and then we apply Equation (4) for three iterations (Lines $6-8$). This process results in the values of recursive dispersion, as defined in [21] and detailed in Section 2.2.3.

Computing the similarities of pairs of links: Our next objective is to come up with the similarity of all pairs of links in the egonet that share a common node. To this end, for every node in the egonet we examine the similarity of all possible pairs of its links (Lines $9-14$). The use of a *min-heap* allows us to maintain the similarities of pairs of links sorted (Line 9). We first calculate the distance of two links using the Jaccard similarity coefficient (Line 12), and then we balance this distance using the previously calculated recursive dispersion measure, as specified in Equation (7). In particular, we divide the value of Jaccard similarity coefficient with the sum of the recursive dispersion of the nodes involved in the links (Line 13). Finally, we insert the resulting similarity value in the heap holding the similarities of all pairs (Line 14).

Creating the dendrogram: The last step of our algorithm is to merge the pairs of links and come up with the dendrogram that portrays the hierarchical structure of the local communities of node u. More specifically, we iterate through our heap holding the sorted similarities of pairs of links, and group the pairs one by one (Lines 15–16). At every grouping stage, we keep track of the action that takes place to allow for the construction of the dendrogram. Moreover, we monitor the current partition density using Equation (5), to determine the best level at which to cut the dendrogram at. When the dendrogram is built, i.e., when we are left with a single cluster, LDLC terminates (Lines 17–18). The resulting dendrogram we come up with reveals the overlapping nature of the network's communities through a rich and intuitive hierarchical structure.

Analysis of LDLC: The running time of our algorithm depends on the calculations needed to come up with the intersection and the union of the neighbors of all pairs of nodes in the egonet. The former is needed for the calculation of both recursive dispersion and similarity, whereas the latter is needed just for calculating similarity. There are totally $\binom{|V_u|}{2}$ pairs of nodes in the egonet, where $|V_u|$ is the number of nodes in the egonet. Both the intersection and union operations require linear time, as we consider representations of sorted adjacency lists. Therefore, the time needed for these calculations is $O(|V_u|^3)$. We maintain an associative array to hold the results for the intersection and union of all pairs of neighbors and access them in constant time in all steps of Algorithm 1. To calculate the recursive dispersion (Lines 6–8) for every neighbor v of u we go through all possible pairs of the common neighbors of u and v, which has a worst case complexity of $O(|V_u|^3)$. For the calculation of the similarities (Lines 9–14) of the pairs of edges with a common neighbor and their placement in a min-heap, the execution time is at worst $O(|V_u|^2 log|V_u|)$. Finally, going through the calculated similarities and joining the pairs of edges (Lines 15–18) requires again $O(|V_u|^2 log|V_u|)$ time. Consequently, the running time of LDLC is $O(|V_u|^3)$. When it comes to egonets of nodes with large degrees, this order may become unmanageable; hence, we continue with a discussion on how we can reduce our search space and be efficient even in such cases.

3.4 Reducing the Search Space

LDLC operates on the egonets of a target node, as community detection in the global structure of the network is prohibitively expensive for large-scale graphs. However, detecting communities in the egonets of certain nodes in the network may have equivalent cost. In particular, many real-world networks, such as the Internet router graph [2], the World Wide Web graph [31], [32], [33], and citation graphs [34], are known to exhibit power law degree distributions and a few of their nodes exhibit extremely large degrees. Therefore, the size of the respective egonets of these nodes is often comparable to the size of the network.

Uncovering the community structure of nodes with large egonets efficiently calls for a sampling technique that is applied on the egonet to reduce the search space. To this end, a straightforward approach is to perform random sampling, i.e., to pick uniformly at random a subset of the nodes in the egonet, and apply LDLC on the respective sub-graph that comprises these nodes. Such an approach would successfully reduce the time needed to execute our algorithm; however, a random sample of the neighborhood of a node exhibiting high degree is likely include many disparate nodes.

We propose an alternative sampling technique, outlined in Algorithm 2. Instead of including nodes in our sample at random, we maintain the most *involved* nodes of the egonet. More specifically, first insert in a min-heap the first k neighbors of a node u (Lines $2-4$). Then, for the rest of the nodes in the egonet (Line 5), we examine whether their degree is larger than that of the inserted node with the smallest degree (Line 6). If so, we remove the node in the root and insert the current node in the min-heap (Line $7-8$). After we have iterated through all nodes in the egonet, the min-heap will hold the nodes with the largest degrees in the network, which is the outcome of Algorithm 2. The min-heap offers constant time access to the inserted node with the smallest degree, as the min-heap holds this node in its root. The insertion operation costs $O(logk)$ whereas the space complexity is $O(k)$.

In our experimental section, we investigate the effectiveness of our sampling technique through a comparison against a random sampling approach, and study the impact of k with regards to both efficiency and effectiveness. We note that communities with more than 100 nodes are reported to be of poor quality [15].

4 Experimental Evaluation

We compare LDLC against three prominent community detection algorithms based on seed-set expansion, namely

Algorithm 2: k-largest(G, u)

input : The egonet of $u \in G$, $G_u = (V_u, E_u)$, and maximum size k.
output : A sample V'_u of V_u, where $N_+(v)$ in the top-k in G_u $\forall v \in V'_u$.

```
begin
    V'_u ← min_heap();
    for i ← 1 to k do
        V'_u ← V_u[i]
    for i ← k + 1 to |V_u| do
        if V_u[i] > V'_u.low() then
            delete V'_u[1];
            V'_u ← V_u[i]
    return V'_u;
```

LEMON [14], LOSP [11], and HeatKernel [12]. All three above algorithms perform *local* community detection and thus, allow for comparison with our approach in a similar setting. We first discuss the specifications of our experimental setting. Then, we proceed with the evaluation of our LDLC by answering the following questions:

- How well does LDLC overcome the need of constraints other methods have, such as requiring multiple seeds to avoid mixing-up multiple overlapping communities, and detecting mostly large communities?
- How well does LDLC perform in detecting communities of real-world networks compared to other methods?
- How efficient is LDLC when compared to other local community detection approaches?
- What is the impact of dispersion-based measures on the quality of the derived hierarchical community structures?
- How effective is our sampling algorithm when compared with a random sampling approach?

4.1 Experimental Setting

Our dataset comprises six social, co-authorship, and co-purchasing networks of different sizes, the details of which are outlined in Section 2.4. We implemented LDLC using Python 2.7 and the Snap.py interface[2] of the SNAP system [26]. Our algorithm is publicly available.[3] We conducted our timing experiments on a Dell PowerEdge R630 server with an Intel®Xeon® E5-2630 v3, 2.40 GHz with 8 cores, and a total of 128GB of RAM. Only one of the CPU cores was used in our experiments.

4.2 Qualitative Evaluation

We begin our discussion on experimental results by illustrating the behavior of our LDLC against LEMON, when discovering the communities of a target node in the *DBLP* co-authorship network.

Figure 5 depicts the egonet of the target node which we use as a seed to both algorithms (white colored node), as well as the communities detected by the two algorithms. The force-directed layout we use to enhance the visualization, reveals that the nodes form two well-connected groups. The nodes of the grouping in the right actually belong to one of *DBLP*'s high quality ground-truth communities to which none of the nodes of the left grouping belongs to. Moreover, we observe, that the pink colored node is part of the left group but features a link with a node that is part of the right group and is not connected with any of the pink node's neighbors other than the white node. This results to a high value of absolute dispersion for the pink node in the egonet of the white node.

We illustrate using grey color part of the community that is detected when providing the white colored node as a seed to LEMON. In particular, LEMON produces a community of 81 nodes in total, featuring *all the neighbors* of the seed node, as well as nodes that are only connected to the target's neighbors. The numbers on the nodes in Figure 5 indicate their ranking according to their likelihood to belong to the target community. We observe that the community detected by LEMON exhibits certain unexpected or undesired attributes. First, high quality ground-truth communities are reported to be much smaller than the community detected by LEMON. In particular, the high quality communities of *DBLP* have an average community size of 22.45 nodes, as shown in Table 1. The community of LEMON however, is more than 3 times as large. Second, using the target node as the single seed results in the participation in the detected community of nodes that belong to different social circles. In particular, LEMON performs random walks starting from the target node to calculate the likelihood of a node belonging to the detected community. Naturally, nodes of different social circles are likely to exhibit high likelihood and LEMON is unable to distinguish between the different and possibly overlapping communities of the target node. This behavior is evident in Figure 5. We observe that nodes ranked from 2 to 7 according to their likelihood, reside in the middle part of the left well-connected group of the seed's neighbors. The node that LEMON adds to the community immediately after, ranked 8th, does not share a single link with these nodes, and clearly belongs to another community. Similarly, LEMON continues to add nodes in the detected community from diverse areas around the seed node, until it meets a stopping criterion. Therefore, we see that LEMON favors nodes that reside in dense areas *regardless of their relevance to one another*. Overcoming this issue would require *multiple cherry-picked* seeds that would increase the likelihood of nodes that are actually part of the same community. This is equally true for other methods that employ random walks for seed set expansion, including LOSP.[4] Last, the pink colored node continues to exhibit high dispersion in the community detected by LEMON, as the community features its link with a node of the cluster on the right side of Figure 5.

Figure 5 also illustrates the communities discovered in the egonet of the white colored node using LDLC. We cut the tree produced by LDLC at the level that produces the optimal partition density and observe that our algorithm detects two communities, depicted with pink and teal color, respectively. The pink community has a size of 12 nodes,

[2]https://snap.stanford.edu/snappy/index.html
[3]https://bitbucket.org/panagiotisl/ldlc

[4]As the authors show in [11] (Figure 2) the presence of three seeds is essential to enable LOSP to distinguish between two overlapping cliques.

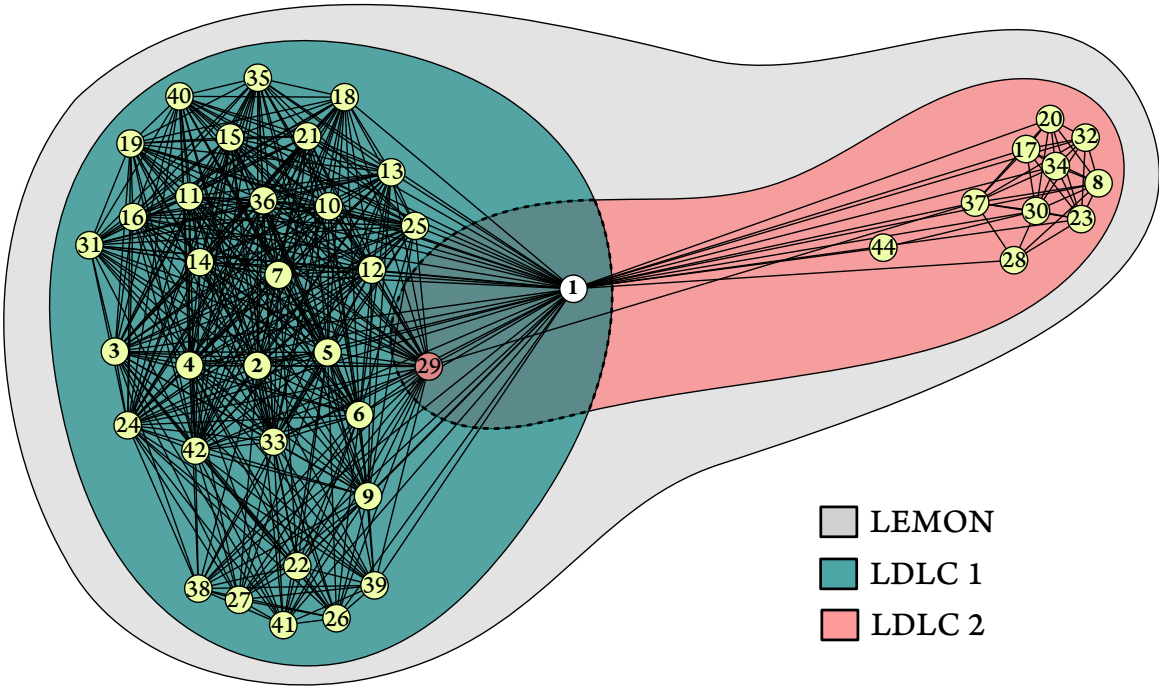

Fig. 5. The egonet of a node in the *DBLP* graph. LEMON's detected community (grey color) features, among others, all the nodes in the egonet. Numbers indicate the LEMON's ranking of the nodes according to their likelihood of belonging to the detected community. LDLC uses hierarchical link clustering in the egonet of the target node and penalizes the links with nodes exhibiting high dispersion to come up with two communities, colored teal and pink.

and the teal community a size of 33 nodes. The average size of the two communities of LDLC (22.5) is very close to the average size of the ground-truth communities of this network. Both detected communities are well-connected. In addition, the pink-colored community is a very accurate detection of an actual ground-truth community. Finally, the pink-colored node is featured in both detected communities and does not exhibit high dispersion in either community.

We saw here that previous approaches may not detect communities well in situations like the one that we described in this qualitative evaluation. Of course, there are other examples where previous approaches can accurately identify communities. Our goal was to show the strengths of our method through a concrete example. To measure performance more objectively, we now turn to comparing the accuracy of previous local community detection techniques and LDLC through the use of our ground truth datasets.

4.3 Evaluation via Ground-Truth

Evaluating and comparing communities detected by different algorithms is not a trivial task. Large networks exhibit extremely complex organization and cannot be visualized in meaningful ways. However, the the presence of networks with ground-truth communities [15] has allowed for a common-ground evaluation context for measuring the accuracy of a community detection algorithm. In particular, recent approaches [11], [12], [14] quantify the similarity of a detected community C and a ground-truth community T using F1 score, which is defined as:

$$F_1(C,T) = \frac{2 * Precision(C,T) * Recall(C,T)}{Precision(C,T) + Recall(C,T)} \quad (8)$$

where precision is the fraction of detected nodes that are relevant and recall is the fraction of relevant nodes that are retrieved:

$$Precision(C,T) = \frac{|C \cap T|}{|C|} \quad (9)$$

$$Recall(C,T) = \frac{|C \cap T|}{|T|} \quad (10)$$

As there is no standard way of selecting a seed, we followed the procedure performed in [12]. We execute LDLC for all ground-truth communities of each network of our dataset, using every single node as an individual seed. For each ground-truth community, we kept the seed that produced the community with the highest F1 score. Table 2 shows the average F1 score of LDLC for all ground-truth communities of each network. In addition, we present results of 3 state-of-the-art local community detection algorithms on the same datasets. In particular, we used the publicly available implementation of LEMON[5] to perform experiments through the same exhaustive procedure. We also executed LEMON using 3 random seeds as suggested in [14]. The results we obtained are worse than the ones reported in [14] for *both* cases, as the optimal initialization setting of LEMON differs for the various networks of our dataset. Therefore, we opt to present in Table 2 the results

[5]https://github.com/YixuanLi/LEMON

TABLE 2
F1 Score comparison.

Algorithm	DBLP	Amazon	Youtube	LiveJournal	Orkut	Friendster
LDLC	**0.843**	0.894	**0.560**	**0.876**	**0.438**	**0.688**
LEMON [14]	0.525	**0.910**	0.190	-	0.170	-
LOSP [11]	0.691	0.845	0.413	0.674	0.216	-
HeatKernel [12]	0.257	0.325	0.177	0.131	0.055	0.078

TABLE 3
Execution time comparison.

Algorithm	DBLP	Amazon	Youtube	LiveJournal	Orkut	Friendster
LDLC	**0.0063** sec	**0.0007** sec	**0.0048** sec	0.1471 sec	0.3742 sec	**0.0642** sec
LEMON	9.2781 sec	9.9206 sec	12.2579 sec	-	13.1432 sec	-
LOSP	0.38 sec	0.04 sec	3.85 sec	1.47 sec	4.74 sec	-
HeatKernel	0.0467	0.01464	0.0714	**0.0353**	**0.04986**	-

reported in [14] instead. We also include the results on the same dataset of LOSP, as reported in [11], and HeatKernel from [12]. We note that the results of LOSP and HeatKernel are obtained using a subset of only 500, and 100 ground-truth communities for each network, respectively.

We see in Table 2 that our LDLC manages to outperform all three state-of-the-art algorithms for all the networks of our dataset, with the exception of the *Amazon* co-purchasing graph for which LEMON is slightly better. The average F1 score of LDLC is significantly larger for all other networks, and the improvement is more evident on the social networks of our dataset, i.e., *Youtube*, *LiveJournal*, *Orkut*, and *Friendster*. For *DBLP*, *Youtube*, and *LiveJournal* the results of LDLC are impressive and much more accurate than all three other methods. Regarding *Orkut*, accurate detection is a particularly hard task, as the size of the ground-truth communities is unusually large in this network. Nonetheless, LDLC is much more effective than the other methods. The friendship graph of *Friendster* almost reaches 2 billion edges, and both LEMON and LOSP have failed to report results for this network due to memory consumption. We are able to operate on the *Friendster* network despite its size, as LDLC employs a memory-efficient representation (SNAP). HeatKernel also manages to report results on graphs of this scale by using pylibbvg.[6] We see in Table 2 that LDLC is able to achieve an F1 score of 0.688, which clearly outperforms HeatKernel. The results regarding the *Amazon* network differentiate due to the particular nature of its communities, which allows all 4 algorithms to achieve their best result regarding accuracy. More specifically, *Amazon* is a co-purchasing network and, thus, does not feature any connector nodes [15]. In addition, the overlapping ground-truth communities of *Amazon* are usually nested communities, that are subsets of other communities [15].

4.4 Execution Time Comparison

We further evaluate LDLC as far as the execution time is concerned. We adopt the methodology followed in citeHeSBHL15 and for every graph of our dataset we execute LDLC for 5,000 trials which consist of choosing a node of the graph uniformly at random to be a seed. We perform the same experiment for LEMON,[7] LOSP,[8] and HeatKernel[9] for the 5 smaller networks of our dataset, as their execution failed for *friendster* in our setting.[10]

We observe that LDLC is able to respond *in real-time* for the communities of all the graphs of our dataset, including *Friendster* that comprises 1,806,067,135 edges. We see in Table 3 that LDLC significantly outperforms both LEMON and LOSP with regard to execution time. This is expected, as LDLC operates only on the egonet of a target node. To produce the egonet we simply need to apply intersection on the sets of neighbors of all neighbors of the target node. Instead, LEMON and LOSP perform multiple random walks to generate a local neighborhood around the target node, a procedure that is much more costly timewise. In addition, the local neighborhood of LEMON or LOSP is usually significantly larger than the egonet of the target node. Therefore, LDLC is applied on a much smaller portion of the original graph, compared to LEMON and LOSP. As far as HeatKernel is concerned, we see in Table 3 that it scales impressively with the size of the network and outperforms all approaches for *LiveJournal* and *Orkut*. However, as Table 2 shows, the accuracy of HeatKernel on larger networks is not comparable to that of the other three approaches.

We note, that the average execution time of LDLC for the *Friendster* graph is smaller than that for *LiveJournal* and *Orkut*, as the egonets of the first are sparser. Thus, LDLC has to iterate over fewer pairs of links in the grouping phase for the graph of *Friendster* and terminates faster.

4.5 Impact of Dispersion on the Resulting Hierarchical Community Structure

LDLC builds on hierarchical link clustering and dispersion-based measures to detect the communities of a single node in its egonet. Having discussed the accuracy and efficiency

[6]https://pypi.python.org/pypi/pylibbvg

[7]https://git.io/fj6Xb
[8]https://git.io/fj6Xy
[9]https://git.io/fj6X5
[10]The largest network that all approaches could handle in our setting is *orkut*. LEMON, LOSP, and HeatKernel required 18.8GB, 8GB, and 4.6GB of memory, respectively. LDLC required 1.7GB.

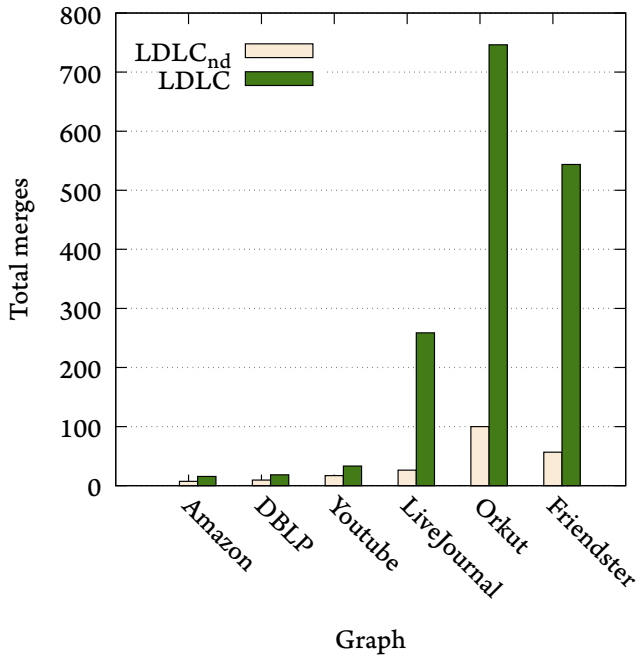

Fig. 6. Impact of the use of recursive dispersion on the number of merges that occur until LDLC terminates. When using recursive dispersion (LDLC) the number of total merges increases significantly. Thus, the resulting dendrogram reveals the hierarchical community structure in greater detail.

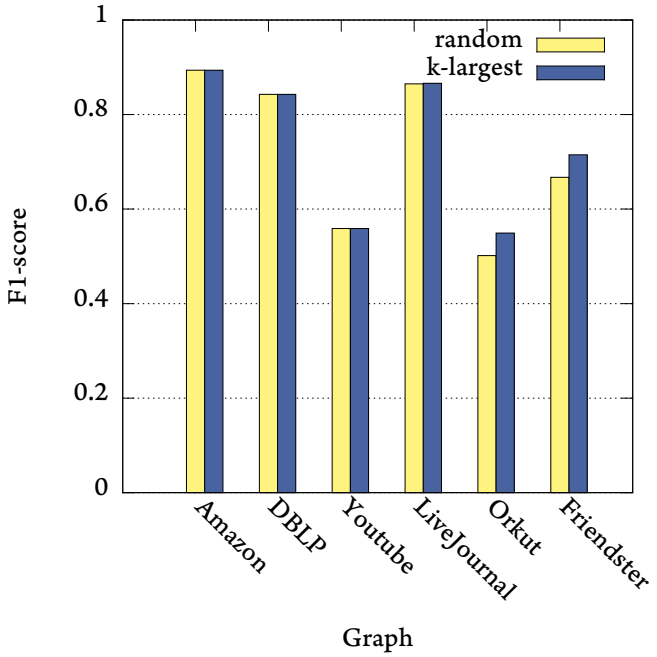

Fig. 7. LDLC results on the networks of our dataset when using a random sampling technique and our k-largest sampling technique. Impact is negligible for the four first networks as very few or no egonets surpass 100 nodes. However, the k-largest sampling technique outperforms the random technique for *orkut* and *friendster*.

of our algorithm we investigate here its effectiveness with regard to deriving a detailed hierarchical community structure. The toy example discussed in Section 3.2 shows that there are cases in which relying strictly on the Jaccard similarity coefficient may result in grouping overlapping communities at an early stage. More specifically, clusters featuring multiple links are likely to be grouped with each other due to the similarity of a single pair at the low levels of the respective dendrogram.

We attempt here to quantify the extent of this trend as well as the impact of recursive dispersion on it. In particular, we consider two settings for LDLC: i) the first one (LDLC$_{nd}$) employs Equation (6) to estimate the similarity of pairs of links, whereas ii) the second one (LDLC) employs Equation (7). Then, we investigate for every ground truth community of every network of our dataset the total merges involving the final cluster, i.e., the one featuring all links of the egonet. We use the number of total merges as a quality function, as it is indicative of the height the dendrogram reaches, and thus, of the detail we achieve with regard to the resulting community structure.

Figure 6 illustrates a comparison between the two settings for all networks of our dataset. We see that the first setting consistently leads to *shorter* dendrograms when compared to the ones resulting using the second setting. Evidently, the use of recursive dispersion results in dendrograms with significantly richer structure. Through Equation (7) LDLC successfully delays the grouping of pairs of links exhibiting high dispersion in the egonet. Thus, our algorithm reveals the hierarchical community structure in greater detail. The impact is noticeable in all networks as we end up with at least twice as much merges using the second setting. The total number of merges depends on network properties such as the average degree and the average community size. For *dblp*, both these properties exhibit small values and thus the number of total merges that occur is small for both settings. In contrast, for *orkut* both these properties exhibit large values and the number of total merges that occur is large for both settings.

We note that, to the best of our knowledge, there exist no real-world or synthetic networks with hierarchically clustered ground-truth communities that would allow for directly evaluating the accuracy of our algorithm as far as the hierarchical structure is concerned.

4.6 Impact of Sampling Technique

We complete our experimental section by evaluating the effectiveness of our sampling technique. More specifically, if the target node has a large egonet LDLC obtains a sample of the egonets and operates on the sample. Algorithm 2 outlines this process in which k nodes exhibiting the largest degrees in the egonet are retrieved efficiently. We examine here the effectiveness of this approach by comparing it with a sampling technique that selects k nodes of the egonet uniformly at random.

Figure 7 illustrates the F1-score results we obtain for all networks of our datasets when applying each of the two techniques for nodes with egonets larger than 100 nodes. There are very few or no such nodes in our results for the four smaller networks of the dataset, i.e., *amazon*, *dblp*, *youtube*, and *livejournal*. Thus, the difference in performance between the two techniques is negligible for these networks. However, for *orkut* and *friendster* several communities included in our results resulted from target nodes with

TABLE 4
Impact of k on our sampling technique.

k	F1-score	Time	Total Merges
200	0.7028	0.0586 sec	493.85
150	0.7095	0.0377 sec	437.44
100	0.7149	0.0207 sec	333.68

egonets comprising more than 100 nodes that were sampled with the use of the two techniques. We observe that using random sampling we achieve an F1-score of 0.50 for *orkut* and 0.67 for *friendster*. The respective values when using k-largest sampling are 0.55 for *orkut* and 0.71 for *friendster*. That is, sampling the most involved nodes of the egonet instead of sampling uniformly at random has a significant impact on the accuracy we achieve.

Furthermore, we investigate the impact of k as far as accuracy, execution time, and total merges are concerned. We focus on *friendster* and perform three experiments using a value of 100, 150, and 200 for k, respectively. Limiting k often leads to a worse recall but can also lead to better precision for LDLC. Therefore, as we see in Table 4, the F1-score is actually improved as we limit k. Moreover, we Table 4 shows noticeable improvements with regard to execution time as we reduce the value of k. Finally, we see that the number of average total merges reduces with k, but remains high even with a value of 100.

5 Related Work

The problem of identifying communities emanates from research on graph partitioning, which has been active since the 1970s [35]. Girvan and Newman, with their seminal paper on community detection [3], build on Freeman's *betweeness centrality* measure [36] and define *edge betweeness* as the number of shortest paths between pairs of vertices that run along an edge. Using this measure, they iteratively remove the edges with high betweeness, as they have a tendency to connect different clusters, and thus, reveal the underlying community structure of a network. The algorithm is computationally expensive, yet this work sparked significant research in the field of community detection [4].

Many clustering methods aim at maximizing *modularity*, a measure introduced by Newman and Girvan [37]. Modularity captures the quality of a specific proposed division of a network into communities, by examining how higher the internal cluster density is than the external cluster density. One such method is that of Clauset et al. [38]. There, the proposed algorithm discovers a hierarchical community structure and identifies the best level to cut the tree at as the one that produces the division that maximizes modularity. Blondel et al. [39] propose Louvain, another greedy modularity maximization algorithm. Nodes are iteratively aggregated into communities as long as such a move locally improves modularity. Methods of this class are know to suffer from a resolution limit [23].

Another popular direction in the field of community detection, is the use of *random walks*. Pons and Latapy [40] use random walks to measure the similarity between vertices. In another line of work, Infomap [41] finds the shortest multilevel description of a *random walker* to get a hierarchical clustering of the network.

The previous methods, hierarchically nested or else, do not take into account the fact that communities in networks may overlap [42]. Palla et al. [42], propose the Clique Percolation Method, a local approach based on *k-cliques*. Overlaps between communities are allowed as a given node can be part of several *k-clique* percolation clusters at the same time. A revolutionary idea in overlapping community detection was introduced in two approaches that were developed almost simultaneously [5], [6]. The core of these approaches is that instead of focusing on grouping nodes, communities should be formed by considering groups of links. This allows for a natural incorporation of overlaps between communities while also retaining a hierarchical community structure. Ahn et al. [5] additionally report a comparison of their proposed algorithm with previous approaches, proving that it outperforms all of them.

Later research efforts focused on providing more scalable approaches. Coscia et al. [43] use *egonet* analysis methods and propose DEMON that allows nodes to vote for the communities they see locally in an effort to improve the quality of overlapping partitions. Yang and Leskovec [9] report that, contrary to previous belief, community overlaps are more densely connected than the non-overlapping parts. This relaxes the assumption that governed all previous efforts on overlapping community detection. Building on their empirical observations, they also propose BigClam [10], a community detection method that uses matrix factorization to detect communities. BigClam requires as an input the number of communities to look for, or else should be guided with the minimum and maximum number of communities as well as the number of tries it should make. Gleich and Seshadhri [7] formalized the problem of community detection as finding vertex sets with small *conductance*, where conductance of a cluster is a measure of the probability that a one-step random walk starting in that cluster, leaves the cluster. They proposed the use of personalized PageRank vectors to identify communities with good conductance scores. A similar approach is investigated in [8], where a number of alternative seeding phases before the use of personalized PageRank vectors is examined. However, minimizing conductance leads to the identification of dense areas of a network as single communities, when they are in fact overlapping parts of multiple communities [25]. These approaches are more efficient than previous overlapping methods but fail to handle massive scale graphs.

Recent approaches depart from the direction of detecting communities on the *global* graph structure. Instead, they detect *local* communities in time functional to the size of the community, and provide support for large scale graphs. Kloster and Gleich [12] propose a deterministic local algorithm to compute heat kernel diffusion and study the communities it produces. The authors compare with PageRank diffusion on real-world datasets and report that their approach is able to detect smaller, more accurate communities, with slightly worse conductance. Li et al. [14] propose LEMON that uses seeds to perform short random walks and form an approximate invariant subspace termed *local spectra*. Then, LEMON looks for the minimum 1-norm vector in the span of this *local spectra* such that the seeds

are in its support. Building on the findings of LEMON, He et al. propose LOSP [11] that is additionally able to detect small communities and performs better when initiated with a single seed. In another line of work, Metwally et al. [44] employ general purpose clustering algorithms to detect click rings that launch advertising traffic fraud attacks. However, their techniques are applicable on *multi-faceted* graphs rather than *single* graphs. Our approach focuses on local communities but employs hierarchical clustering of pairs of links in the egonet of a target node, using *tie strength* measures that effectively handle networks with dense overlapping parts of communities. Thus, we efficiently reveal a more accurate hierarchical community structure in large scale networks.

A preliminary version of our work appeared in [45]. In this extended version:

- We delve into the merits of employing dispersion-based measures with regard to the hierarchical community structure provided by our algorithm. We show that such measures result in richer hierarchical structures through experimentation on our entire dataset.
- We propose a sampling technique that reduces the search space of our algorithm by focusing on the most *involved* nodes of an egonet. This technique allows us to maintain the efficiency of our algorithm in cases when target nodes exhibit large degrees in the network.
- We compare our sampling technique against a random sampling technique and show that our approach is very effective.
- We carry out the entire range of our experimentation using our sampling technique and report updated results with regards to both accuracy and execution time.

In another line of work [46], [47], we apply community detection via seed set expansion on graph-streams. Our approaches allow for local community detection without a need for in-memory representation of the network, as we employ a one-pass streaming approach and effectively determines the size of communities automatically. However, these approaches require more seed nodes than LDLC.

6 Conclusion

In this paper, we propose and develop LDLC, a novel local community detection algorithm for large scale graphs. LDLC focuses on the egonet of a target node in the network and performs hierarchical agglomerative clustering on the egonet's pairs of links. We investigate measures that evaluate the strength of ties in networks, building on the notion that mutual neighbors of nodes may be or may be not well interconnected. The nodes involved in ties that belong in the second category, act as connector nodes between overlapping communities. Therefore, in a hierarchical approach they should be considered for grouping when the higher levels of the respective dendrogram are forming. We achieve that, by using the recursive dispersion measure to balance the similarity of two links and prioritize the grouping of pairs of links with mutual neighbors that function in a single context. Consequently, our approach is able to handle overlapping communities appropriately and provides increased accuracy, while also revealing the rich hierarchical structure of the communities of a node in the network. We compare LDLC with three state-of-the-art local community detection methods to highlight the effectiveness of our approach when handling overlapping areas of multiple communities. Moreover, we examine the accuracy of all algorithms against ground-truth communities and find that LDLC significantly outperforms all of them for a wide range of publicly available networks. Our timing experiments showcase that LDLC additionally offers improved efficiency and scales to large scale graphs. Finally, we discuss the merits of employing dispersion-based measures, as well as applying a sampling technique we introduce on the egonets of target nodes.

We believe that an interesting future direction would be a drift from the currently available ground-truth communities depicting metadata groups [48] to communities that better portray the functional roles of a network's nodes. Such communities will allow for a more accurate comparison of community detection techniques. To this end, we will collect data from social network groups where membership signifies affinity.

Acknowledgments

The authors would like to express their gratitude towards the anonymous reviewers for providing insightful comments and valuable feedback. A preliminary version of our work appeared in [45].

References

[1] I. de Sola Pool and M. Kochen, "Contacts and influence," *Social networks*, vol. 1, no. 1, pp. 5–51, 1978.
[2] M. Faloutsos, P. Faloutsos, and C. Faloutsos, "On power-law relationships of the internet topology," in *ACM SIGCOMM computer communication review*, vol. 29, no. 4. ACM, 1999, pp. 251–262.
[3] M. Girvan and M. E. Newman, "Community structure in social and biological networks," *Proc. of the National Academy of Sciences*, vol. 99, no. 12, pp. 7821–7826, 2002.
[4] S. Fortunato, "Community detection in graphs," *Physics Reports*, vol. 486, no. 3, pp. 75–174, 2010.
[5] Y.-Y. Ahn, J. P. Bagrow, and S. Lehmann, "Link communities reveal multiscale complexity in networks," *Nature*, vol. 466, no. 7307, pp. 761–764, 2010.
[6] T. Evans and R. Lambiotte, "Line graphs, link partitions, and overlapping communities," *Physical Review E*, vol. 80, p. 016105, 2009.
[7] D. F. Gleich and C. Seshadhri, "Vertex neighborhoods, low conductance cuts, and good seeds for local community methods," in *Proc. of the 18th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, 2012, pp. 597–605.
[8] J. J. Whang, D. F. Gleich, and I. S. Dhillon, "Overlapping community detection using seed set expansion," in *Proc. of the 22nd ACM Int. Conf. on Information & Knowledge Management*, 2013, pp. 2099–2108.
[9] J. Yang and J. Leskovec, "Community-affiliation graph model for overlapping network community detection," in *Proc. of the 12th IEEE International Conference on Data Mining*, 2012, pp. 1170–1175.
[10] ——, "Overlapping community detection at scale: a nonnegative matrix factorization approach," in *Proc. of the 6th ACM int. Conf. on Web Search and Data Mining*, 2013, pp. 587–596.
[11] K. He, Y. Sun, D. Bindel, J. E. Hopcroft, and Y. Li, "Detecting overlapping communities from local spectral subspaces," in *IEEE International Conference on Data Mining, Atlantic City, NJ, USA*, 2015, pp. 769–774.
[12] K. Kloster and D. F. Gleich, "Heat kernel based community detection," in *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA*, 2014, pp. 1386–1395.
[13] I. M. Kloumann and J. M. Kleinberg, "Community membership identification from small seed sets," in *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*, 2014, pp. 1366–1375.

[14] Y. Li, K. He, D. Bindel, and J. E. Hopcroft, "Uncovering the small community structure in large networks: A local spectral approach," in *Proc. of the 24th Int. Conf. on World Wide Web*, 2015, pp. 658–668.
[15] J. Yang and J. Leskovec, "Structure and overlaps of ground-truth communities in networks," *ACM Transactions on Intelligent Systems and Technology*, vol. 5, no. 2, p. 26, 2014.
[16] D. F. Gleich and M. W. Mahoney, "Mining large graphs," in *Handbook of Big Data*, ser. Handbooks of modern statistical methods, P. Bühlmann, P. Drineas, M. Kane, and M. van de Laan, Eds. CRC Press, 2016, pp. 191–220.
[17] M. S. Granovetter, "The strength of weak ties," *American journal of sociology*, pp. 1360–1380, 1973.
[18] P. V. Marsden and K. E. Campbell, "Measuring tie strength," *Social forces*, vol. 63, no. 2, pp. 482–501, 1984.
[19] S. L. Feld, "The focused organization of social ties," *American journal of sociology*, pp. 1015–1035, 1981.
[20] D. H. Felmlee, "No couple is an island: A social network perspective on dyadic stability," *Social Forces*, vol. 79, no. 4, pp. 1259–1287, 2001.
[21] L. Backstrom and J. Kleinberg, "Romantic partnerships and the dispersion of social ties: A network analysis of relationship status on facebook," in *Proc. of the 17th ACM Conf. on Computer Supported Cooperative Work & Social Computing*, 2014, pp. 831–841.
[22] Newman, M. E.J., "Detecting community structure in networks," *Eur. Phys. J. B*, vol. 38, no. 2, pp. 321–330, 2004. [Online]. Available: https://doi.org/10.1140/epjb/e2004-00124-y
[23] S. Fortunato and M. Barthélemy, "Resolution limit in community detection," *Proceedings of the National Academy of Sciences*, vol. 104, no. 1, pp. 36–41, 2007.
[24] J. Yang and J. Leskovec, "Defining and evaluating network communities based on ground-truth," in *Proc. of the 12th IEEE International Conference on Data Mining*, 2012, pp. 745–754.
[25] ——, "Overlapping communities explain core–periphery organization of networks," *Proc. of tihe IEEE*, vol. 102, no. 12, 2014.
[26] "Stanford Network Analysis Project," https://snap.stanford.edu/.
[27] P. Boldi and S. Vigna, "The webgraph framework I: compression techniques," in *Proc. of the 13th Int. Conf. on World Wide Web, New York, NY, USA, May 17-20*, 2004, pp. 595–602.
[28] P. Liakos, K. Papakonstantinopoulou, and M. Sioutis, "Pushing the Envelope in Graph Compression," in *Proc. of the 23rd ACM Int. Conf. on Information and Knowledge Management*, Shanghai, China, 2014, pp. 1549–1558.
[29] P. Liakos, K. Papakonstantinopoulou, and A. Delis, "Memory-optimized distributed graph processing through novel compression techniques," in *Proc. of the 25th ACM Int. Conf, on Information and Knowledge Management, CIKM 2016, Indianapolis, IN, USA, October 24-28, 2016*, pp. 2317–2322.
[30] ——, "Realizing memory-optimized distributed graph processing," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 4, pp. 743–756, April 2018.
[31] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *science*, vol. 286, no. 5439, pp. 509–512, 1999.
[32] A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J. Wiener, "Graph structure in the web," *Computer networks*, vol. 33, no. 1, pp. 309–320, 2000.
[33] L. A. Adamic and B. A. Huberman, "Power-law distribution of the world wide web," *science*, vol. 287, no. 5461, pp. 2115–2115, 2000.
[34] S. Redner, "How popular is your paper? an empirical study of the citation distribution," *The European Physical Journal B-Condensed Matter and Complex Systems*, vol. 4, no. 2, pp. 131–134, 1998.
[35] B. W. Kernighan and S. Lin, "An efficient heuristic procedure for partitioning graphs," *Bell system technical journal*, vol. 49, no. 2, pp. 291–307, 1970.
[36] L. C. Freeman, "A set of measures of centrality based on betweenness," *Sociometry*, pp. 35–41, 1977.
[37] M. E. J. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Phys. Rev. E*, vol. 69, no. 2, p. 026113, Feb. 2004.
[38] A. Clauset, M. E. Newman, and C. Moore, "Finding community structure in very large networks," *Physical review E*, vol. 70, no. 6, p. 066111, 2004.
[39] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2008, no. 10, p. P10008, 2008.
[40] P. Pons and M. Latapy, "Computing communities in large networks using random walks," in *Computer and Information Sciences-ISCIS 2005*, 2005, pp. 284–293.
[41] M. Rosvall and C. T. Bergstrom, "Multilevel compression of random walks on networks reveals hierarchical organization in large integrated systems," *PloS one*, vol. 6, no. 4, p. e18209, 2011.
[42] G. Palla, I. Derényi, I. Farkas, and T. Vicsek, "Uncovering the overlapping community structure of complex networks in nature and society," *Nature*, vol. 435, no. 7043, pp. 814–818, 2005.
[43] M. Coscia, G. Rossetti, F. Giannotti, and D. Pedreschi, "DEMON: a local-first discovery method for overlapping communities," in *Proc. of the 18th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, 2012, pp. 615–623.
[44] A. Metwally, J. Pan, M. Doan, and C. Faloutsos, "Scalable community discovery from multi-faceted graphs," in *IEEE Int. Conf. on Big Data, Santa Clara, CA, USA*, 2015, pp. 1053–1062.
[45] P. Liakos, A. Ntoulas, and A. Delis, "Scalable link community detection: A local dispersion-aware approach," in *2016 IEEE International Conference on Big Data, BigData 2016, Washington DC, USA, December 5-8, 2016*, 2016, pp. 716–725.
[46] ——, "CoEuS: Community detection via seed-set expansion on graph streams," in *2017 IEEE International Conference on Big Data, BigData 2017, Boston, MA, USA, December 11-14, 2017*, 2017, pp. 676–685.
[47] P. Liakos, K. Papakonstantinopoulou, A. Ntoulas, and A. Delis, "DiCeS: Detecting communities in network streams over the cloud," in *12th IEEE Int Conf. on Cloud Computing, CLOUD 2019, Milan, Italy*, 2019.
[48] D. Hric, R. K. Darst, and S. Fortunato, "Community detection in networks: Structural communities versus ground truth," *Physical Review E*, vol. 90, no. 6, p. 062805, 2014.

Panagiotis Liakos is a Postdoctoral researcher at the University of Athens, where he obtained his Ph.D. on distributed and streaming graph processing techniques. His research interests include graph mining and information retrieval, with a particular focus on mining large scale graphs and streams of social activity.

Alexandros Ntoulas is an Assistant Professor of Computer Science at the University of Athens. He holds both a Ph.D. and an M.Sc. in Computer Science from the University of California, Los Angeles and a Diploma in Computer Engineering from the University of Patras. He has received a best paper award (ICDE 2005), a best paper runner-up award (WWW 2009), and a best applied data science reviewer award (KDD 2017).

Alex Delis is a Professor of Computer Science at the University of Athens and New York University Abu Dhabi. His research interests are in Distributed and Virtualized Data Systems. His research work has been supported by agencies and organizations in Australia, US and the European Union. He holds both a Ph.D. and an M.Sc. in Computer Science from the University of Maryland at College Park and a Diploma in Computer Engineering from the University of Patras. He is a member of the IEEE Computer Society, the ACM, and the Technical Chamber of Greece.