LinkBlackHole*: Robust Overlapping Community Detection Using Link Embedding

Jungeun Kim, Sungsu Lim, Jae-Gil Lee, and Byung Suk Lee

Abstract—This paper proposes LinkBlackHole*, a novel algorithm for finding communities that are (i) overlapping in nodes and (ii) mixing (not separating clearly) in links. There has been a small body of work in each category, but this paper is the first one that addresses both. LinkBlackHole* is a merger of our earlier two algorithms, LinkSCAN* and BlackHole, inheriting their advantages in support of highly-mixed overlapping communities. The former is used to handle overlapping nodes, and the latter to handle mixing links in finding communities. Like LinkSCAN and its more efficient variant LinkSCAN*, this paper presents LinkBlackHole and its more efficient variant LinkBlackHole*, which reduces the number of links through random sampling. Thorough experiments show superior quality of the communities detected by LinkBlackHole* and LinkBlackHole to those detected by other state-of-the-art algorithms. In addition, LinkBlackHole* shows high resilience to the link sampling effect, and its running time scales up almost linearly with the number of links in a network.

Index Terms—Community Detection, Graph Clustering, Overlapping Communities, Link Clustering, Graph Drawing, Link Embedding

1 Introduction

Community detection in real-world networks such as social networks, collaboration networks, citation networks, and communication networks warrants that individuals may belong to multiple communities, e.g., families, friends, colleagues, and schools [37]. Thus, recently there has been increasing emphasis on *overlapping* community detection [1], [6], [15], [22], [33], whereby each individual can belong to more than one communities. Overlapping community detection is known to be harder than disjoint community detection [33], [48]. Currently, LinkSCAN* [33], which introduced the novel concept of *link-space transformation*, has been regarded as the state-of-the-art algorithm. It transforms the original graph to a link-space graph, where each node is mapped from a link (i.e., edge) in the original graph in such a way that two nodes are adjacent if the corresponding links in the original graph share a common end node. The benefit of the transformation is that *disjoint* community detection from the link-space graph produces *overlapping* communities.

Overlapping community detection is further complicated by another issue: *mixing* of links between communities. Quantitatively, mixing is defined as the fraction of links in the network that are crossing between different communities [24]. Such links ambiguate the detection of boundaries between communities and consequently make community structures harder to detect. Lim et al. [32] showed that clustering accuracy of the existing community detection algorithms indeed degrades drastically as the mixing increases. This study was done in the context of disjoint community detection, but the adverse effect of mixing is observed to be worse for overlapping community detection, as it is confounded by overlapping nodes.

Lim et al. [32] resolved the mixing problem by incorporating a *geometric embedding* technique in their algorithm BlackHole, which we call the *black hole transformation* henceforth. It is modified from the graph drawing which is a well-known technique used for aesthetic visualization of a graph. Multiple nodes that are likely to belong to the same community are mapped to (almost) the same position (dubbed a "black hole") as a result of the embedding. Clustering is then performed on the resulting positions.

One intuitive idea for achieving robust overlapping community detection despite the mixing effect is to combine (1) the link-space transformation [33] and (2) the black hole transformation [32]. We propose this combined algorithm and call it *LinkBlackHole**. The algorithm is essentially *link* embedding that is done by BlackHole when it works on the link-space graph output by LinkSCAN* because the nodes embedded are mapped from links in the original graph. It may generate too many unnecessary overlaps once the nodes in the communities are mapped back to links in the original graph (details in Section 4.4). This problem has been resolved by adopting the *belonging coefficient* used in a node-based community detection method, e.g., the label propagation algorithm [15].

Figure 1 summarizes the merit of link embedding in LinkBlackHole*, inherited from the link-space transformation of LinkSCAN* and the black hole transformation (geometric embedding) of BlackHole. Concretely, the link-space transformation enables us to accurately find *overlapping* communities in the original graph through finding *disjoint* communities in the link-space graph while preserving the original graph structure. In turn, the black hole

- *J. Kim and J.-G. Lee are with the Graduate School of Knowledge Service Engineering, KAIST, Daejeon 34141, Republic of Korea. E-mail: {je_kim, jaegil}@kaist.ac.kr*
- *S. Lim is with the Department of Computer Science and Engineering, Chungnam National University, Daejeon 34134, Republic of Korea. E-mail: sungsu@cnu.ac.kr*
- *B. Lee is with the Department of Computer Science, University of Vermont, Burlington, VT 05405, USA. E-mail: bslee@uvm.edu*

Corresponding author: Jae-Gil Lee.

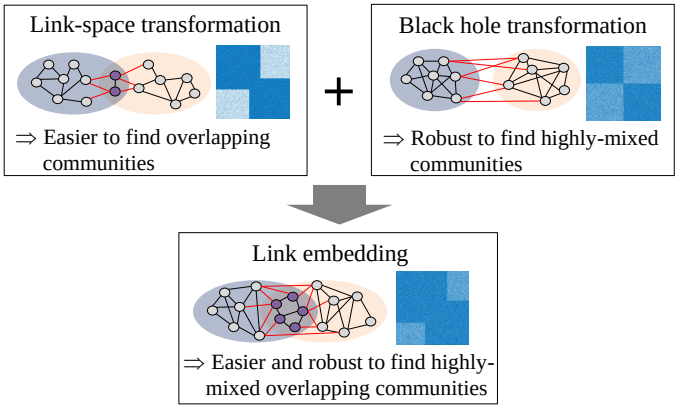


Fig. 1: The advantage of link embedding in LinkBlackHole*.

transformation achieves robust accuracy of finding *highly-mixed* overlapping communities.

In this paper, we first present LinkBlackHole, a robust overlapping community detection algorithm based on link embedding, and then its variant LinkBlackHole*, which improves the clustering efficiency of LinkBlackHole by reducing the number of links through random sampling while maintaining comparable clustering accuracy. This algorithm development structure parallels that of LinkSCAN and LinkSCAN* [33].

LinkBlackHole* has four main steps, as illustrated in Figure 2. First, the original graph $\mathcal{G}$ is converted to a link-space graph $LS(\mathcal{G})$ by the link-space transformation, and the links of $LS(\mathcal{G})$ are sampled. (Please note that this sampling is not part of LinkBlackHole.) Second, every node in the sampled link-space graph $LS'(\mathcal{G})$ is mapped to a point in a low dimensional space. Third, the positions are clustered using a conventional clustering algorithm such as DBSCAN [10], and link-space communities in $LS'(\mathcal{G})$ are determined from the clusters. Fourth, the link membership in $LS'(\mathcal{G})$ is translated back to node membership in $\mathcal{G}$.

Contributions made through this paper are as follows.

1. We propose a novel *link embedding* framework that incorporates the advantages of both the link-space transformation and the black hole transformation.
2. We present a robust algorithm LinkBlackHole for *highly-mixed overlapping* community detection and its enhanced algorithm LinkBlackHole* that achieves *higher efficiency with negligible error* compared with LinkBlackHole.
3. We empirically show that LinkBlackHole* achieves higher accuracy than the state-of-the-art algorithm LinkSCAN* and others, especially for detecting highly-mixed overlapping communities.

The content of this paper is an extension of our two prior papers [32], [33] that constitute the bases of the work. The extensions made are mainly in (1) proposing *link embedding*, which integrates the link-space transformation [33] and the black hole transformation [32], to achieve high-quality community detection from highly-mixed and overlapping networks structures; (2) solving the problem of excessively overlapping communities when the two transformations are integrated into one, by way of adopting the belonging coefficient [15]; (3) significantly extending the literature review and conducting a new set of experiments to evaluate LinkBlackHole* in conjunction with LinkBlackHole. In addition, the contents adopted from the two base papers [32], [33] have been completely rewritten to fit the objective and scope of this paper.

The rest of this paper is organized as follows. Section 2 discusses related work. Section 3 provides key preliminary background helpful to understand the rest of the paper. Section 4 discusses our proposed overlapping community detection algorithm LinkBlackHole*. Section 5 presents the evaluations and their results. Section 6 concludes the paper.

2 Related Work

Many algorithms have been proposed for overlapping community detection. Xie et al. [48] conducted an extensive survey on this topic. Representative algorithms in their survey are based on five classes of approaches: clique percolation, link partitioning, local expansion, fuzzy detection, and agent-based [48]. Let us discuss them briefly here. We will also discuss two additional categories: network property-based and egonet-based. Note that an algorithm may belong to more than one category.

The *clique percolation* is density-based. It builds each community by forming k-cliques and enlarging them through merging to maximize the modularity [50] or percolation by union if, for example, two k-cliques share $k-1$ nodes [22], [23]. Thus, for a given k, the members in a community obtained by this approach can be reached through connected subsets of nodes, and the communities may share nodes with each other. This approach has the inherent problem of being sensitive to the parameter k, which affects the trade-off between the quality of clustering and the coverage depending on the value of k. Specifically, when k is large, it tends to find high-quality communities, but the communities cover only a small fraction of nodes (i.e., lower coverage).

The *link-partitioning* is based on the similarity among *relationships* [1], [11]. It first converts the graph $\mathcal{G}$ to a line graph $L(\mathcal{G})$, where each node of $L(\mathcal{G})$ represents a link of $\mathcal{G}$ and two nodes of $L(\mathcal{G})$ are adjacent if and only if their corresponding links share a common node in $\mathcal{G}$. Then, it can easily show that the node-partition of $L(\mathcal{G})$ has one-to-one correspondence to the link-partition of $\mathcal{G}$, which is an overlapping clustering for a set of nodes. However, this approach tends to find unnecessary small clusters, and too many nodes overlap between clusters.

The *local expansion* is based on local optimization [6]. It first finds initial seeds based on a certain seeding strategy, such as random nodes [24], high degree nodes [3], [47], centroids of disjoint pre-clusters [47], and maximal cliques [26], [43]. Then, it expands from the seeds to form clusters, mostly using a certain local fitness benefit function as the fitness function. OSLOM [25] is somewhat unique in that it uses the *statistical significance* of community structure for fitness. All these local expansion algorithms greedily expand seeds until the fitness function is locally optimized. Their performances, therefore, are highly dependent on initial seeds, which is problematic.

The *fuzzy detection* is based on measuring the strength of association between nodes and communities using a membership vector, which denotes a belonging factor. This approach covers such algorithms as spectral clustering using

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2018.2873750, IEEE Transactions on Knowledge and Data Engineering

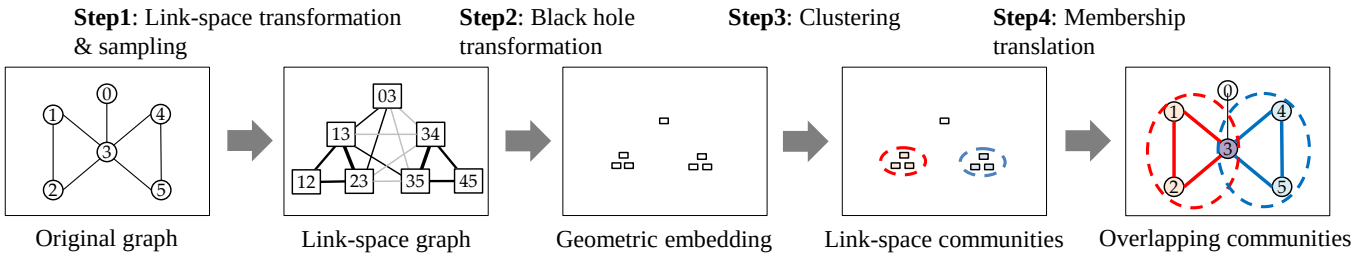


Fig. 2: The main steps of LinkBlackHole*.

fuzzy C-means clustering [51], nonlinear constrained optimization with fuzzy node membership [34], non-negative matrix factorization using Bayesian inference to matrix partitioning [39], and mixture models built as generative models for edges [41] or nodes [13] in the network. However, this approach requires the selection of the dimensionality k of the membership vector, which is not feasible to determine before community detection.

The *agent-based* approach includes the label propagation algorithm [15], general speaker-listener based information propagation framework [49], and game-theoretic framework [4]. Based on our observation, the representative one is the label propagation algorithm, where all nodes propagate their labels to their neighbors for one step so that they reach a consensus on their community membership in many cases. For overlapping community detection, it extends the label and propagate steps to include the information about more than one community [15]. Owing to the simplicity of the procedure, the label propagation method is very fast, but it fails to converge in general [17].

Network properties are used by some algorithms to provide a clue to clustering. CONGA [14] utilizes a network property called *betweenness* to split nodes so that overlapping clusters are possible. ModuLand [21] introduces a *centrality*-type property based on the influence of a node on a link and uses it in clustering. These algorithms suffer from very high computational cost, hence generally are not considered effective.

An *egonet* in the network is a subgraph made of groups, where each group consists of a node and its neighbors and the links among them. Overlapping communities are formed by merging these groups [40]. A more recent work in this category is *ego-splitting* [9], which constructs local egonets using local expansion and splits them through node replications to allow for overlapping clusters and then performs global partitioning on the resulting network.

3 Preliminaries

In this section, we give an overview of basic techniques needed by the two base algorithms, LinkSCAN* [33] and BlackHole [32]. The notations used in this paper are summarized in Table 1.

3.1 Link Clustering

A community of *links* is known to have a theoretical advantage over a community of *nodes* in that a link is likely to have a unique identity whereas a node tends to have multiple identities [20]. Thus, recently, there has been a few community detection studies based on link clustering [1], [11], [46], formally defined in Definition 1.

TABLE 1: Summary of the notation.

Notation	Description
$\mathcal{G}$	an original undirected weighted graph
$LS(\mathcal{G})$	the link-space graph of $\mathcal{G}$
v_i	a node in an original graph
e_{ij}	a link in an original graph
$v_{e_{ij}}$	a node in a link-space graph
k_v	the degree of a vertex v
w_v	the weight of a vertex v
$w_{u,v}$	the weight of an edge $\{u, v\}$
$\sigma(\cdot,\cdot)$	similarity between two links in $\mathcal{G}$
$\omega(\cdot,\cdot)$	similarity between two nodes in $LS(\mathcal{G})$
$p(v)$	the position of a vertex v by a graph layout p
$\mathcal{E}(p\|\mathcal{G})$	the energy with a graph layout p for $\mathcal{G}$
a, r	the parameters of the (a, r)-energy model
$\mathcal{B}$	a set of black holes
$\mathcal{LC}$	a set of link-space communities in $LS(\mathcal{G})$
$\mathcal{C}$	a set of overlapping communities in $\mathcal{G}$

Definition 1. Let $\mathcal{G} = (V, E)$ be a given network. Then, a collection, $\mathcal{L}$, of subsets of E is said to be a *link clustering* of $\mathcal{G}$ if the elements of $\mathcal{L}$ are pairwise disjoint and the union of the elements of $\mathcal{L}$ is equal to E.

Most notably, Ahn et al. [1] proposed the link-partition method, which uses a Jaccard-type similarity score between links. The membership of a node was determined by those of its incident links. Note that a node participates in multiple communities if it has multiple incident links with different types of relationship.

Link clustering of a graph $\mathcal{G}$ is translated to graph partitioning of the *line graph* $L(\mathcal{G})$ that represents the structure among the links of $\mathcal{G}$. Specifically, each link of $\mathcal{G}$ is mapped a node of $L(\mathcal{G})$, where two nodes of $L(\mathcal{G})$ are adjacent if the corresponding links of $\mathcal{G}$ *intersect* at some node in $\mathcal{G}$. With this mapping scheme, however, $L(\mathcal{G})$ gives too much prominence to high-degree nodes of the original graph $\mathcal{G}$ because a node of degree k_i in $\mathcal{G}$ is mapped to form a clique of size k_i, which has $k_i(k_i - 1)/2$ links, in $L(\mathcal{G})$ [11].

3.2 Graph Drawing

For a given graph $\mathcal{G} = (V, E)$, *graph drawing* (or *graph layout*) produces a node-link diagram that pictorially represents the topological structure (i.e., vertices and edges) of the graph in a two-dimensional drawing space. A *layout* is a function $p : V \mapsto S$ that maps from vertices in V to vertex positions in a drawing space $S(= \mathbb{R}^2)$. The traditional graph drawing approaches aim to produce graphs that are aesthetic, symmetric, and crossing-free in their layouts according to their own layout strategies. Among these approaches, we adopt the *force-directed layout* approach, as it is simple and intuitive [45], and theoretically strong [27].

The force-directed layout approach is based on the physical metaphor of *attractive force* and *repulsive force* in mechanical springs or molecular mechanics [8], [12], [35]. The metaphor is that (1) there is the attractive force between adjacent vertices connected via a spring and (2) there is the repulsive force between vertices (electrically charged particles). Thus, adjacent vertices attract to form a group of densely-connected vertices, and all pairs of vertices repulse to separate sparsely-connected vertices.

Formally, for a layout p and two vertices u and v ($u \neq v$), the *attractive force* exerted on u by v is defined by Eq. (1), and the *repulsive force* exerted on u by v is defined by Eq. (2) [36]. The strengths of the forces are often chosen to be proportional to some power of the distance.

$$\text{Attractive force}: \quad w_{u,v}\|p(u)-p(v)\|^{a}\overrightarrow{p(u)p(v)} \tag{1}$$

$$\text{Repulsive force}: \quad w_u w_v\|p(u)-p(v)\|^{r}\overrightarrow{p(v)p(u)} \tag{2}$$

Here, $\|p(u)-p(v)\|$ denotes the distance between u and v, and $\overrightarrow{p(u)p(v)}$ denotes a unit-vector pointing from $p(u)$ to $p(v)$.

The approach finds an equilibrium state such that the net force on each vertex becomes zero. It is achieved by minimizing the *energy* defined in Definition 2.

Definition 2. [36] The *energy* $\mathcal{E}(p|\mathcal{G})$ of a layout p for a graph $\mathcal{G}=(V,E)$ is given by Eq. (3). Thus, graph drawing is solved as an optimization problem with the objective function in Eq. (3) in a search space of all possible layouts (i.e., p's).

$$\mathcal{E}(p|\mathcal{G}) = \sum_{\{u,v\}\in E} w_{u,v}\frac{\|p(u)-p(v)\|^{a+1}}{a+1} - \sum_{\{u,v\}\in V^{(2)}} w_u w_v \frac{\|p(u)-p(v)\|^{r+1}}{r+1} \tag{3}$$

Noack [36] has studied a general framework for the force-directed layout approach. There, the energy model in Eq. (3) is called the (a,r)*-energy model*. This model is used in several well-known traditional graph drawing algorithms, such as the Fruchterman-Reingold algorithm [12] (with $a=2, r=-1$), the Davidson-Harel algorithm [8] (with $a=1, r=-3$), and the LinLog algorithm [35] (with $a=0, r=-1$). Note that these algorithms have different layout characteristics depending on the values of the two parameters a and r.

4 The LinkBlackHole* Framework

This section discusses the LinkBlackHole*, the framework developed for robust overlapping community detection, in conjunction with LinkBlackHole. As mentioned in Section 1, the framework has four major steps, outlined in Algorithm 1. Recall that LinkBlackHole has no link sampling done in Step 1, and otherwise the same as LinkBlackHole*.

4.1 Step 1: Link-Space Transformation and Sampling

4.1.1 Link-Space Transformation

As stated in Section 1, the link-space transformation converts a graph to a link-space graph. Definition 3 shows three characteristics required of the transformation.

Algorithm 1 LinkBlackHole/LinkBlackHole*

INPUT: a graph $\mathcal{G}=(V,E)$
OUTPUT: a set of overlapping communities C found in $\mathcal{G}$;
1: /* Step 1. Link-Space Transformation and Sampling (§4.1) */
2: $LS(\mathcal{G})=(V_{LS},E_{LS}) \leftarrow$ Link-Space Transformation($\mathcal{G}$);
3: $LS(\mathcal{G}) \leftarrow$ Link Sampling($LS(\mathcal{G})$,α,β); /* LinkBlackHole* only */
4: /* Step 2. Black Hole Transformation (§4.2) */
5: $a \leftarrow -0.95$, $r \leftarrow -1.0$; /* Embedding params (§4.2.1) */
6: $\mathcal{B} \leftarrow$ Geometric Embedding($LS(\mathcal{G}), a, r$); /* (§4.2.2) */
7: /* Step 3. Clustering (§4.3) */
8: ε,*MinPts* $\leftarrow$ Estimate Param($\mathcal{B}$);/* Clustering params */
9: $\mathcal{BC} \leftarrow$ DBSCAN($\mathcal{B}, \varepsilon,$ *MinPts*); /* Clustering */
10: **for each** $\mathcal{BC}_j \in \mathcal{BC}$ **do**
11: /* Compose a community from clusters */
12: $\mathcal{LC}_j \leftarrow$ Set of links from a black hole cluster $\mathcal{BC}_j$;
13: $\mathcal{LC} \leftarrow \mathcal{LC} \cup \{\mathcal{LC}_j\}$;
14: **end for**
15: /* Step 4. Membership Translation (§4.4) */
16: $\mathcal{C} \leftarrow$ Membership Translation($\mathcal{LC}$, $LS(\mathcal{G})$, τ);
17: **return** $\mathcal{C}$

Definition 3. Given a graph $\mathcal{G}$, its *link-space graph* $LS(\mathcal{G})$ satisfies the following properties.

- A node $v_{e_{ij}}$ in $LS(\mathcal{G})$ represents the link e_{ij} between the nodes v_i and v_j in $\mathcal{G}$.
- Two nodes, $v_{e_{ik}}$ and $v_{e_{jk}}$, in $LS(\mathcal{G})$ are adjacent if and only if their corresponding links share a common end node, v_k, in $\mathcal{G}$.
- The weight $\omega(v_{e_{ik}}, v_{e_{jk}})$ on the link between $v_{e_{ik}}$ and $v_{e_{jk}}$ is assigned by a similarity function $\sigma(e_{ik}, e_{jk})$ calculated on $\mathcal{G}$.

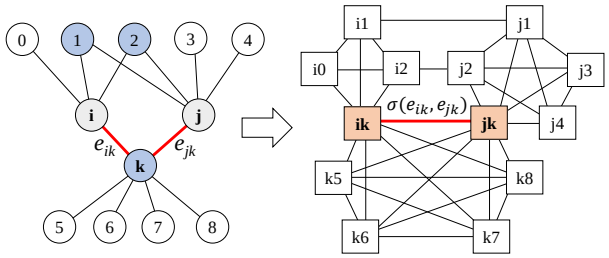
Regarding the link similarity $\sigma(\cdot,\cdot)$ in Definition 3, a simple yet effective Jaccard-type measure proposed by Ahn et al. [1] (see Section 3.1) is used. (Other measures, such as those in [31], could be used as well.) The link similarity $\sigma(\cdot,\cdot)$ is formally defined as Eq. (4).

$$\omega(v_{e_{ik}}, v_{e_{jk}}) = \sigma(e_{ik}, e_{jk}) = \frac{|\Gamma(v_i)\cap\Gamma(v_j)|}{|\Gamma(v_i)\cup\Gamma(v_j)|} \tag{4}$$

Here, $\Gamma(v)$ is $\{u : d(v,u) \leq 1\}$, and $d(v,u)$ is the length of a shortest path between v and u. This similarity measure provides a natural way to calculate the similarity between two incident links since it is free of the degree of the common end node v_k.

Example 1. Figure 3 shows the link-space transformation from the original graph $\mathcal{G}$ to the link-space graph $LS(\mathcal{G})$. Two links e_{ik} and e_{jk} in $\mathcal{G}$ are mapped to two nodes $v_{e_{ik}}$ and $v_{e_{jk}}$ in $LS(\mathcal{G})$ since they share a common node v_k in $\mathcal{G}$. The weight of the link $\sigma(e_{ik}, e_{jk})$ is assigned using the structure of $\mathcal{G}$ as follows: $\Gamma(v_i) = \{v_0, v_1, v_2, v_i, v_k\}$ and $\Gamma(v_j) = \{v_1, v_2, v_3, v_4, v_j, v_k\}$; then, $\Gamma(v_i)\cap\Gamma(v_j) = \{v_1, v_2, v_k\}$ and $\Gamma(v_i)\cup\Gamma(v_j) = \{v_0, v_1, v_2, v_3, v_4, v_i, v_j, v_k\}$; hence, the similarity score is $\sigma(e_{ik}, e_{jk}) = \frac{3}{8} = 0.375$. □

Algorithm 2 outlines the steps of the link-space transformation. The links in $\mathcal{G}$ are transformed to the nodes in $LS(\mathcal{G})$ (Lines 1∼4). The nodes in $LS(\mathcal{G})$ are connected



(a) Original graph. (b) Link-space graph.

Fig. 3: An illustration of constructing a link-space graph.

Algorithm 2 Link-Space Transformation

INPUT: a graph $\mathcal{G} = (V, E)$
OUTPUT: the link-space graph $LS(\mathcal{G}) = (V_{LS}, E_{LS})$

1: /* Create the nodes of $LS(\mathcal{G})$ */
2: **for** each $(v_x, v_y) \in E$ **do**
3: insert $v_{e_{xy}}$ into V_{LS};
4: **end for**
5: /* Create the links of $LS(\mathcal{G})$ */
6: **for** each $v_z \in V$ **do**
7: create $N(v_z) = \{v_w | d(v_z, v_w) = 1\}$;
8: **for** each $v_{w_1} \in N(v_z)$ **do**
9: **for** each $v_{w_2} \in N(v_z) \backslash \{v_{w_1}\}$ **do**
10: insert $(v_{e_{zw_1}}, v_{e_{zw_2}})$ into E';
11: **end for**
12: **end for**
13: **end for**
14: /* Assign a weight to each link of $LS(\mathcal{G})$ */
15: **for** each $(v_{e_{zw_1}}, v_{e_{zw_2}}) \in E'$ **do**
16: $\omega(v_{e_{zw_1}}, v_{e_{zw_2}}) = \frac{|\Gamma(v_{w_1}) \cap \Gamma(v_{w_2})|}{|\Gamma(v_{w_1}) \cup \Gamma(v_{w_2})|}$;
17: **end for**
18: **return** $LS(\mathcal{G}) = (V_{LS}, E_{LS})$ /* the link-space graph of $\mathcal{G}$ */

directly if the corresponding links in $\mathcal{G}$ are incident to a common end node (Lines 5∼13). Weights are calculated using the links in $\mathcal{G}$ and are assigned to each link in $LS(\mathcal{G})$ (Lines 14∼17).

Theorem 1. The time and space complexity of Algorithm 2 is $O(|E| + |V|\langle k^2 \rangle)$, where $\langle k^2 \rangle$ is the average of the square of the degree of the $\mathcal{G}$.

Proof 1.

Time complexity: Generating the nodes of a link-space graph $LS(\mathcal{G})$ takes $O(|E|)$ at Lines 1–4, and generating the links of $LS(\mathcal{G})$ takes $O(|V|\langle k^2 \rangle)$ at Lines 5–13, because every pair of incident links should be checked for each node.

Space complexity: The space complexity is the sum of the storage requirement for $\mathcal{G}$ and that for $LS(\mathcal{G})$. The former is $O(|V| + |E|)$, and the latter is $O(|E| + |V|\langle k^2 \rangle)$ [33]. This completes the proof. □

This transformation algorithm as such takes into account two types of graphs—the original graph (G) and the link-space graph ($LS(G)$)—and thereby enables finding *overlapping* communities using a *non*-overlapping community detection algorithm while preserving the structure of the original graph. In a sense, this idea resembles that of the kernel trick [16], where the support vector machine (SVM) attempts to find a separating plane in a higher-dimensional feature space while the kernel function is evaluated against the original data points.

4.1.2 Link Sampling

As noted already, link sampling applies to LinkBlackHole* only. One inherent effect of the link-space transformation is that the number of links resulting from the transformation can be excessive. Since the running time of LinkBlackHole is heavily dependent on the number of links, it would be reasonable to reduce the computational overhead by reducing the number of links considered in the link-space graph $LS(\mathcal{G}) = (V_{LS}, E_{LS})$. Thus, for each node $v \in V_{LS}$, LinkBlackHole* takes a random sample from the set of incident links of v and, consequently, produces a smaller network $LS'(\mathcal{G}) = (V_{LS}, E'_{LS})$, where $E'_{LS} \subset E_{LS}$.

Determining the sample size, n_v, is certainly a key issue to achieving an adequate performance. LinkBlackHole* determines the sample size n_v of each node v using Eq. (5).

$$n_v = min\{k_v, \alpha + \beta \ln k_v\} \tag{5}$$

Here, k_v is the degree of the node v, and α and β are non-negative constants used to control the sample size. If the degree k_v of a node is smaller than $\alpha + \beta \ln k_v$, all incident links are included in the sample. This sampling technique has proven to incur an insignificant error even when the sample size n_v is small. For details of the proof, readers are referred to Theorem 2 (based on Chernoff bound [5], [33]) in the LinkSCAN* paper [33].

What is important in determining the values of the control parameters (α, β) is to achieve high computational efficiency while maintaining low error due to the sampling. In this regard, the *average degree* of nodes has been chosen to be the most relevant factor [28]. The average degree in a link-space graph, $\langle k' \rangle$, is computed as in Eq. (6), where $\langle k \rangle$ is the average degree in the original graph. Under the assumption that the distribution of degrees is not skewed, $\langle k' \rangle$ can be approximated as $2\langle k \rangle - 2$.

$$\langle k' \rangle = \frac{2\sum_{v \in V} k_v(k_v - 1)/2}{\sum_{v \in V} k_v/2} = 2(\langle k^2 \rangle / \langle k \rangle - 1) \approx 2\langle k \rangle - 2 \tag{6}$$

In the case of LinkSCAN* [33], it performs well at $\alpha = 2\langle k \rangle$ and $\beta = 1$, but in the case of LinkBlackHole*, high accuracy is achieved even at a smaller sample size thanks to the high clustering tendency of the black hole transformation. Extensive experiments indeed show that LinkBlackHole* achieves high accuracy even at a much smaller sample size (i.e., when $\alpha = \frac{1}{2}\langle k \rangle$ and $\beta = 1$) than LinkSCAN*.

4.2 Step 2: Black Hole Transformation

This step uses the geometric embedding technique from graph drawing, adopted from the BlackHole robust community detection algorithm [32]. With this technique, each vertex in the sampled link-space graph is mapped to a position in a 2-dimensional drawing space. (The dimensionality of drawing space may well be higher than 2, albeit finite.)

A position in the graph drawing space to which *multiple* vertices are mapped is called a *black hole* as in Definition 4.

Definition 4. Given a graph $\mathcal{G} = (V, E)$ and mapping of V to a set of positions P in a graph drawing space, a *black*

hole $B_i \in P$ is a position to which two or more vertices, $V_{B_i} \subseteq V$, are mapped. For practical purpose, positions that are only slightly different in their coordinates are merged to the same position through truncation of their coordinates. The set of all black holes found this way is denoted by $\mathcal{B}$.

4.2.1 Model Selection

The layout of a graph drawn varies greatly depending on the graph drawing model and, therefore, the choice of the model is important to find a layout of points suitable for community detection. For this choice, the (a, r)-energy model (see Section 3.2), which is a general model used in many graph drawing and has been used successfully in BlackHole, is used in LinkBlackHole* as well. The layout of a graph drawn under this model depends on the values of a and r, therefore, it is necessary to find their optimal values. For this purpose, LinkBlackHole* uses the same optimal values determined by BlackHole, $a = -0.95$ and $r = -1.0$, explained below.

Regarding the value of $a = -0.95$, ideally all vertices belonging to the same community in a graph should be mapped to a single position in the drawing space. Thus, analogously to real black holes in the outer space, the attractive force should become stronger as connected vertices become closer to each other in the drawing space. As shown in Figure 4, when a approaches -1.0 the attractive force grows stronger exponentially as the distance between vertices approaches 0. In addition, $a \geq -1.0$ is required [36]. Thus, $a = -0.95$ proves to be a suitable value.

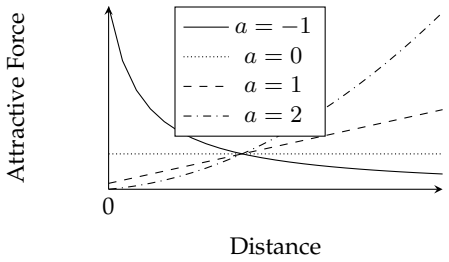


Fig. 4: Attractive force depending on a. (This figure shows how the attractive force changes with the distance between points in the drawing space for different values of a under the (a, r)-energy model.)

Regarding the value of $r = -1.0$, it is the typical value of r used in several popular force-directed graph layout algorithms such as LinLog [35], ForceAtlas [18], and Fruchterman-Reingold [12]. r must be smaller than a, and distances between vertices are less dependent on densities for large $a - r$ [36]. Since a large negative value of r makes $a - r$ large and the value of r should be smaller than $a(= -0.95)$, $r = -1.0$ is the right value.

For further validation of the values chosen for a and r, the clustering tendency of the resulting black holes was examined for varying a with r fixed to -1.0. Since our graph drawing can be considered micro clustering [52], the clustering boundaries are less ambiguous when clustering tendency is higher. Figure 5 shows the Hopkins statistic, which measures the clustering tendency of a data set [2]. The closer it is to 1, the higher the clustering tendency is. We generated a set of 24 various graphs by changing the parameters of the LFR benchmark and, for each value of a, measured the Hopkins statistic of the layout results for the set of the graphs. In the box plot of Figure 5, the Hopkins statistic tends to increase as a approaches -1.0 and becomes almost 1 when $a = -0.95$. Overall, this examination confirms that our model selection with $a = -0.95$ and $r = -1.0$ is indeed correct for our purposes.

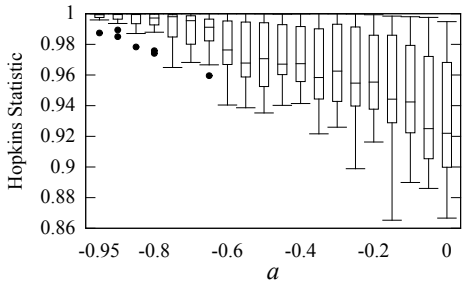


Fig. 5: Hopkins statistic for different values of a.

Thus, the graph drawing model used in LinkBlackHole* is to minimize the energy expressed as Eq. (7), which is derived from Eq. (3) in Section 3.2 by setting $a = -0.95$ and $r = -1.0$ and by transforming $\|p(u) - p(v)\|^{-1+1}/(-1+1)$ into $\ln \|p(u) - p(v)\|$ because $\frac{d}{dx}(\ln x) = \frac{1}{x}$.

$$\mathcal{E}(p|\mathcal{G}) = \sum_{\{u,v\} \in E} w_{u,v} \|p(u) - p(v)\|^{0.05} \times 20 - \sum_{\{u,v\} \in V^{(2)}} w_u w_v \ln \|p(u) - p(v)\| \quad (7)$$

4.2.2 Geometric Embedding Algorithm

Algorithm 3 shows the geometric embedding by graph drawing, which places each vertex in a 2-dimensional space and determines a set of black holes. The algorithm first distributes the vertices randomly in the drawing space (Lines 1∼4) and then keeps updating their positions toward a lower energy (Lines 5∼25). More specifically, in each iterative step, it builds a quadtree of given positions (Lines 7∼8), calculates the attractive and repulsive forces exerted on each vertex using the quadtree (Lines 9∼20), and then changes to positions that have lower energy (Lines 21∼24). In the actual implementation, the repulsive force expressed in Eq. (2) is divided by the total weight of all vertices in order to prevent vertices from spreading too much in the drawing space (Line 15).

Calculating the repulsive energy requires calculating the distance between every pair of vertices. A straightforward approach would take $\mathcal{O}(|V|^2)$, which is quite expensive. To reduce this cost, the algorithm approximates the calculation using a quadtree (Lines 8, 14∼16). Specifically, the drawing space is recursively partitioned into quadrants until a quadrant satisfies the following stop condition: $s/d < \theta$, where s is the quadrant's side length, d is the distance between the quadrant's centroid and the entire drawing space's centroid, and θ is a certain threshold value. Then, all positions in the same quadrant at the leaf level are approximated to the centroid of those positions, and the centroid is given the sum of their weights. The approximation error was formally proven in the context of the n-body problem [42]. Thus, it is easier to set the stop condition using the quadtree than using other

Algorithm 3 Geometric Embedding

INPUT: An undirected weighted graph $\mathcal{G} = (V, E)$
Parameter values of a and r
OUTPUT: A set of black holes $\mathcal{B} = \{p(v)|v \in V\}$

/* Generate initial positions of vertices */
for each $v \in V$ **do**
  $p(v) \leftarrow$ *Unif*$([-0.5, 0.5] \times [-0.5, 0.5])$;
end for
/* Minimize the energy by moving positions */
repeat
  /* Construct a quadtree for a layout */
  $T \leftarrow$ *quadtree*$(\{p(v)|v \in V\})$;
  /* Compute the net force acting on each vertex */
  for each $v \in V$ **do**
    for each u such that $\{u, v\} \in E$ **do**
      $\overrightarrow{f}^{(a)}(v) \leftarrow \overrightarrow{f}^{(a)}(v)+$ Eq. (1); /* attractive */
    end for
    for each leaf $R \in T$ **do**
      $\overrightarrow{f}^{(r)}(v) \leftarrow \overrightarrow{f}^{(r)}(v)+$ Eq. (2); /* repulsive */
    end for
    $\overrightarrow{f}(v) \leftarrow \overrightarrow{f}^{(a)}(v) + \overrightarrow{f}^{(r)}(v)$; /* net force */
  end for
  /* Choose step size γ to minimize $\mathcal{E}(p|\mathcal{G})$ (Eq. (7)) */
  $\gamma \leftarrow argmin_{\gamma \in \{2^{-i}|i=0,\ldots,6\}} \mathcal{E}(p + \gamma \overrightarrow{f}|\mathcal{G})$;
  /* Determine new positions. */
  for each $v \in V$ **do**
    $p(v) \leftarrow p(v) + \gamma \overrightarrow{f}(v)$;
  end for
until energy is not decreasing
return $\mathcal{B} = \{p(v)|v \in V\}$ /* the set of black holes */

structures such as the R-tree. This approximation reduces the required number of distance calculations significantly to $\mathcal{O}(|V| \log |V|)$ [32].

Theorem 2. When Algorithm 3 is run against $LS(\mathcal{G})$, the time complexity is $O(|V|\langle k^2\rangle + |E| \log |E|)$, and the space complexity is $O(|E| + |V|\langle k^2\rangle)$, where $\langle k^2\rangle$ is the average of the square of the degree of the $\mathcal{G}$.

Proof 2.
Time complexity: In a link-space graph $LS(\mathcal{G})$, the number of nodes becomes $|E|$, and the number of links becomes $|V|\langle k^2\rangle$. The time complexity of Algorithm 3 is linear to the number of links plus quasilinear ($n \log n$) to the number of nodes [32].
Space complexity: The space complexity is the sum of the storage requirement for $LS(\mathcal{G})$ and that for the quadtree for the nodes of $LS(\mathcal{G})$. The former is $O(|E| + |V|\langle k^2\rangle)$ [33], and the latter is $O(|E|)$ because the space complexity of the quadtree is linear to the input size [38]. This completes the proof. □

4.3 Step 3: Clustering

The set of black holes, $\mathcal{B}$, determined in Step 2 is passed to a clustering algorithm to generate a set of clusters of black holes, $\mathcal{BC}$. Each cluster of black holes, $\mathcal{BC}_j$ $(j = 1, 2, \ldots) \in \mathcal{BC}$, represents a *link-space community* specified in Definition 5.

Definition 5. A *link-space community* $\mathcal{LC}_j$ $(j = 1, 2, ...)$ is a set of vertices in the link-space graph $LS(\mathcal{G})$ that were mapped to the black holes in $\mathcal{BC}_j \in \mathcal{BC}$. That is, $\mathcal{LC}_j = \{V_{B_i} \mid B_i \in \mathcal{BC}_j\}$ where $V_{B_i} (\subset V_{LS})$ denotes the set of vertices mapped to the black hole B_i in Step 2.

Note that each black hole belongs to only one link-space community, i.e., $\mathcal{BC}_i \cap \mathcal{BC}_j = \emptyset$ for $i \neq j$. The set of all link-space communities, $\mathcal{LC}$, is composed of $\mathcal{LC}_j$'s $(j = 1, 2, ...)$ as specified in Lines 10∼14 of Algorithm 1.

Practically any clustering algorithm can be used to generate $\mathcal{BC}$, and LinkBlackHole* uses DBSCAN [10], which is one of the representative density-based clustering algorithms. DBSCAN is particularly suitable to meet two conditions inherent in LinkBlackHole*: the number of clusters is not known in advance, and the shape of a cluster is not necessarily circular.

DBSCAN$(\mathcal{B}, \varepsilon, \textbf{\textit{MinPts}})$ and **Estimate Param**$(\mathcal{B})$: DBSCAN needs two input parameters—the size (i.e., distance) of a cluster (ε) and the minimum required number of points in a cluster (*MinPts*). LinkBlackHole* determines their values following the heuristic suggested by the authors of DBSCAN. That is, *MinPts* is set to approximately twice the dimensionality, thus 5 for a 2-dimensional space. Then, the value of ε is estimated by plotting the distance to the (*MinPts*-1)$^{\text{th}}$ nearest neighbor for each of sampled points, sorted in descending order, and finding the distance to an "elbow" of the curve.

Note that the clustering in this step is performed on the black hole positions in the graph drawing space, and the resulting clusters ($\mathcal{BC}$) are mapped (one-to-one) back to *node* clusters in the link-space graph $LS(\mathcal{G})$, which are *link-space* communities in Definition 5. The resulting set of all link-space communities, $\mathcal{LC}$, is passed to Step 4.

4.4 Step 4: Membership Translation

This step identifies the membership of a node translated from the result of clustering ($\mathcal{LC}$) in Step 3. A straightforward approach would be to map the community membership of a link to those of its end nodes such that a node that has multiple incident links belongs to multiple communities. However, this approach has a strong tendency to produce unnecessary memberships with weak ties (i.e., sparse connections) because of the high clustering tendency of the black hole transformation. That is, for highly-mixed community structures, every edge that lies between two communities can be incorrectly assigned and, consequently, unnecessary highly-mixed overlapping communities can be formed. It has been observed that preventing such unnecessary memberships can significantly improve the performance of community detection. LinkBlackHole* does it by adopting the *belonging coefficient*, which is used in node-based clustering methods for overlapping community detection, e.g., label propagation [15].

Definition 6. For a vertex v_i and a community C_k, the *belonging coefficient* of v_i in C_k, $bc(v_i, C_k)$, denotes the strength of v_i's membership in C_k and is calculated as the fraction of v_i's incident edges that belong to C_k.

Membership Translation($\mathcal{LC}$, $LS(\mathcal{G})$, τ**)**: Using the belonging coefficient, the membership translation can be implemented straightforwardly as specified in Definition 7. With a threshold value τ, the membership translation assigns a vertex to a community if its belonging coefficient is

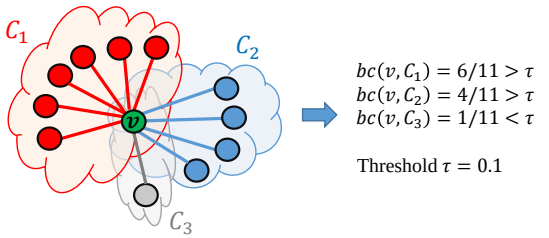


Fig. 6: An example of membership translation.

larger than the threshold, i.e., if more than a cutoff fraction of its incident edges have the same membership.

Definition 7. For a vertex v_i, the *membership translation* assigns v_i to a set $C(v_i)$ of communities such that $C(v_i) = \{C_k \,|\, bc(v_i, C_k) > \tau\}$, where $bc(v_i, C_k)$ is the belonging coefficient of v_i in C_k and τ is a given threshold.

Figure 6 illustrates the membership translation. The community membership C_3 is not assigned to the node v because $bc(v, C_3) < \tau$. Note that the straightforward approach mentioned above is equivalent to this membership translation with $\tau = 0$.

The optimal value of the threshold τ may vary depending on the characteristics of network configuration. As in Figure 7, the simulation showed that the network density (i.e., the average degree $\langle k \rangle$) has the dominant effect on determining the optimal belonging coefficient. This is because a higher average degree of nodes means overall more incident links to individual nodes, and it results in an increase of overlapping nodes connected to weak ties. In addition, as shown in Figure 7a, the optimal value of τ is approximately equal to $\langle k \rangle \times 0.01$. In contrast, other configuration parameters (e.g., μ, O_n, and O_m) have little or no effect on determining the optimal value of τ, as shown in Figures 7b, c, and d.

5 EVALUATIONS

The performances of LinkBlackHole and LinkBlackHole* were evaluated through extensive experiments. In Sections 5.1 and 5.2, these two algorithms are compared with five state-of-the-art overlapping community detection algorithms in terms of the quality of community output. The algorithms compared are listed below. (They are referred to using the acronyms in the parentheses.)

- LinkBlackHole (LB)
- LinkBlackHole* (LB*)
- LinkSCAN* (LS): Link-space transformation + SCAN [33]
- COPRA (CO): Label propagation method [15]
- CPM (CPM): Clique percolation method [22]
- LinkPartition (LP): Link clustering method [1]
- DEMON (DEM): Local expansion method [6]

In Section 5.3, the effect of link sampling on LinkBlackHole* is examined in terms of the community quality and running time compared with LinkBlackHole. In Section 5.4, the running time scalability of LinkBlackHole* is compared with those of the five state-of-the-art algorithms.

All experiments were done on Ubuntu Linux Servers with one CPU (Intel Xeon Processor E5-2670) and 96 GBytes of main memory. LinkBlackHole and LinkBlackHole* were implemented in Java. For all other algorithms, we used the software packages provided by the authors. LinkSCAN* was implemented in C++, COPRA in Java, CPM in Python, LinkPartition in C++, and DEMON in Python.

5.1 Community Quality for Synthetic Networks

5.1.1 Data Sets

Synthetic networks were generated by the Lancichinetti-Fortunato-Radicchi (LFR) benchmark [23]. LFR benchmark networks are generated with a built-in community structure that resembles the features found in most real-world networks with power degree distribution [23], and are widely used for comparing community detection algorithms [19], [30], [48]. The LFR benchmark also allows for rich flexibility in controlling the network topology by tuning network configuration parameters. Table 2 lists the parameters used in this experiment. The networks size N = 5000 is the most common setting for evaluating overlapping community detection algorithms [30], [33], [48]. O_n and O_m are two parameters[1] that collectively reflect how the communities overlap in their nodes overall.

For each setting of the parameters, ten network instances were generated, and the performance was measured by taking an average of repeating the same experiment against the ten network instances.

The belonging coefficient cutoff threshold τ (see Section 4.4) was adjusted to its empirical optimal depending on the setting of the network density parameter $\langle k \rangle$, which is dominantly influential on τ.

5.1.2 Performance Measure

The *normalized mutual information (NMI)* [7] was chosen to measure the quality of output communities, i.e., the accuracy of community detection. The NMI is an information-theoretical measure that compares the memberships of two sets and is widely used for community detection. (One of the two sets is the set of communities found by the algorithm, and the other is the set of ground-truth communities.) Its value ranges from 0 to 1, and a higher value means better quality.

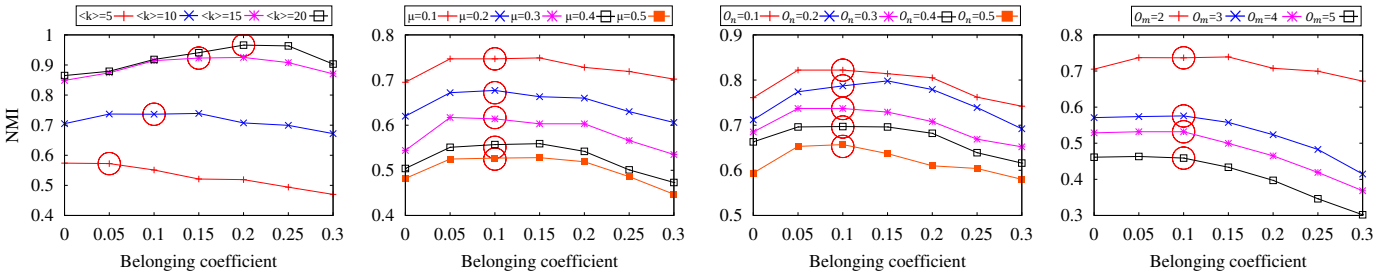
5.1.3 Results

We discuss the accuracy results in three aspects of the network, for varying (1) network density ($\langle k \rangle$), (2) degree of community mixing (μ), and (3) degree of community overlap (O_n) and community membership count of a node (O_m), while fixing the others parameters to the default values shown in Table 2.

Varying network density: Figure 8 shows the effect of the average degree $\langle k \rangle$, varying from 5 to 25, on the clustering accuracy for two different values of com mixing (μ), 0.1 and 0.3. For all algorithms, the accuracy increases as the average degree increases. This result is natural from the fact that overlapping community structures are harder to detect for sparse networks [33], [48]. LinkBlackHole and LinkBlackHole* perform better than all the other algorithms

1. The original LFR benchmark defines O_n as the *number* of overlapping nodes, but we redefine it as the *fraction* of overlapping nodes for ease of exposition.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2018.2873750, IEEE Transactions on Knowledge and Data Engineering



(a) For different values of $\langle k \rangle$. (b) For different values of μ. (c) For different values of O_n. (d) For different values of O_m.

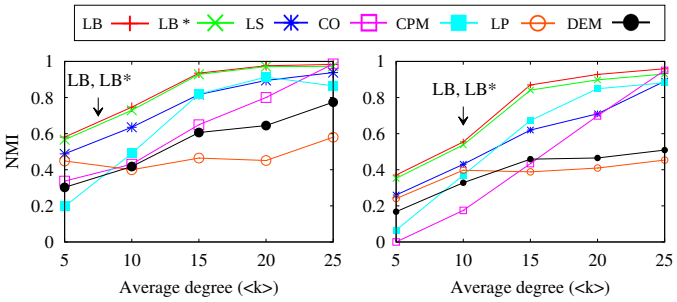
Fig. 7: Effects of the belonging coefficient on the quality of output community for different values of network configuration parameters. (See Table 2 in Section 5.1.1 for the parameters and their default values.)

TABLE 2: Network configuration parameters for the LFR benchmark.

Symbol	Measure	Definition	Short name	Default
N	network size	the number of nodes	node count	5000
$\langle k \rangle$	network density	the average degree of all nodes	average degree	10
μ	degree of community mixing	the fraction of edges crossing different communities among all edges	com mixing	0.1, 0.3
O_n	degree of community overlap	the fraction of nodes belonging to multiple communities among all nodes	com overlap	0.3
O_m	community membership count of a node	the number of communities a node belongs to	membership count	2

in the entire range of $\langle k \rangle$ regardless of the degree of community mixing (μ). Notably, the performance advantage holds strong even when the network is sparse (e.g., for $\langle k \rangle \leq 15$), specifically, achieving 1.14 to 2.96 (1.53 on average) times higher accuracy when $\mu = 0.1$ and 1.29 to 5.69 (1.95 on average) times higher accuracy when $\mu = 0.3$.

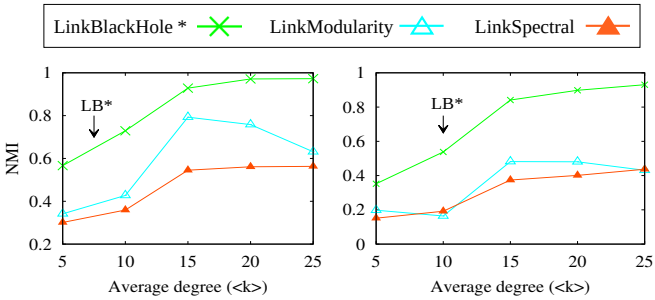
Examining the other algorithms a bit further, it is observed that LinkSCAN* shows the second best performance overall for sparse networks (e.g., when $\langle k \rangle \leq 10$); COPRA and CPM show low accuracy in sparse networks but show increasingly competitive accuracy as the network becomes denser (e.g., when $\langle k \rangle \geq 20$); LinkPartition and DEMOM consistently perform poorly in the entire range of $\langle k \rangle$.



(a) Com mixing (μ) = 0.1. (b) Com mixing (μ) = 0.3.

Fig. 8: Effect of the average degree of a node ($\langle k \rangle$).

Varying a clustering algorithm for the link-space graph: In order to verify that the link-space transformation and the black hole transformation are a good match, we replaced the BlackHole algorithm in LinkBlackHole* with modularity optimization (Louvain) and spectral clustering, which were the second best algorithms in our earlier work [32]. These new variations are denoted by *LinkModularity* and *LinkSpectral*. Figure 9 shows the result of these variations. LinkBlackHole* significantly outperforms both LinkModularity and LinkSpectral, achieving 1.17 to 2.03 (1.64 on average) times higher accuracy when $\mu = 0.1$ and 1.75 to 3.28 (2.26 on average) times higher accuracy when $\mu = 0.3$. Thus, this result indeed shows the validity of the merger of our two earlier algorithms.



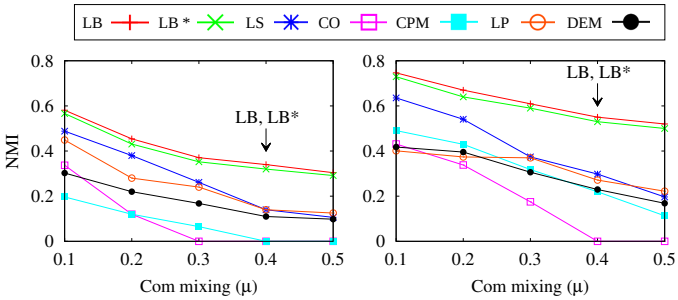
(a) Com mixing (μ) = 0.1. (b) Com mixing (μ) = 0.3.

Fig. 9: Effect of a clustering algorithm for the link-space graph in the experiment of Figure 8.

Varying the degree of community mixing: Figure 10 shows the effect of the com mixing (μ), varying from 0.1 to 0.5, on the clustering accuracy for two different values of the average degree ($\langle k \rangle$), 5 and 10. For all algorithms, the accuracy decreases as the degree of community mixing increases, which is a natural result from the fact that discovering *overlapping* communities becomes harder as the communities are mixed to a higher degree [33], [48]. LinkBlackHole and LinkBlackHole* outperform the other algorithms in the entire range of the com mixing, specifically, outperforming the next best algorithm (LinkSCAN*) by 1.18 to 2.39 times as μ changes from 0.1 to 0.5 and overall achieving 1.19 to 5.4 (3.03 on average) times higher accuracy than all the other algorithms when $\langle k \rangle = 5$ and 1.17 to 4.59 (2.31 on average) times when $\langle k \rangle = 10$. Notably, the performance advantage becomes more prominent as the com mixing increases. This result demonstrates the superior robustness of LinkBlackHole and LinkBlackHole* to the com mixing, which blurs the community boundaries.

Varying degree of community overlap and membership count of a node: Figures 11 and 12 show the effects of com overlap (O_n) varying from 0.1 to 0.5 and membership count (O_m) varying from 2 to 5, respectively, for two differ-

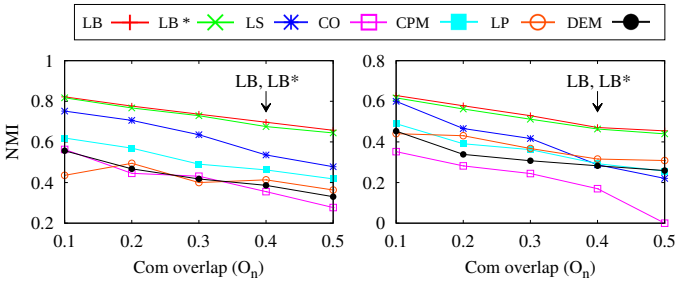
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2018.2873750, IEEE Transactions on Knowledge and Data Engineering



(a) Average degree ($\langle k \rangle$) = 5. (b) Average degree ($\langle k \rangle$) = 10.

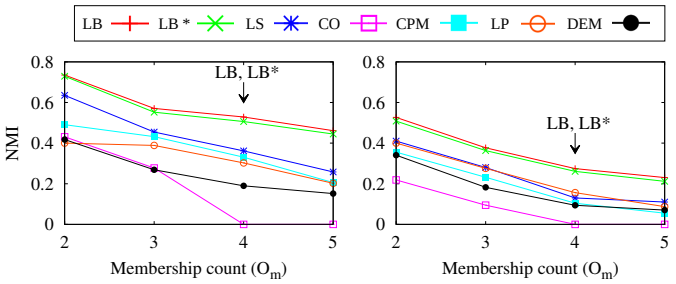
Fig. 10: Effect of the degree of community mixing (μ).

ent values of com mixing (μ), 0.1 and 0.3. For all algorithms, the accuracy decreases as the com overlap and membership count increase, which is from the fact that discovering overlapping community structures is harder when there are more overlapping communities in the network [33], [48]. Here again, LinkBlackHole and LinkBlackHole* are the best performers, and outperform the next best algorithm (LinkSCAN*) 1.12 times when $O_n = 0.1$ or $O_m = 2$ and 1.68 times when $O_n = 0.5$ or $O_m = 5$, overall achieving 1.09 to 3.04 times higher accuracy than all the other algorithms when $\mu = 0.1$ and 1.05 to 4.18 times when $\mu = 0.3$.



(a) Com mixing (μ) = 0.1. (b) Com mixing (μ) = 0.3.

Fig. 11: Effect of the degree of community overlap (O_n).



(a) Com mixing (μ) = 0.1. (b) Com mixing (μ) = 0.3.

Fig. 12: Effect of the community membership count of a node (O_m).

Summary of synthetic network experiment results: LinkBlackHole and LinkBlackHole* achieved higher accuracy than the other algorithms in both highly-mixed networks (Figure 10) and highly overlapping networks (Figures 11 and 12). In addition, in every case of increasing com mixing (μ), com overlap (O_n), and membership count (O_m), the performance gap between the proposed algorithms and the other algorithms increased. These results demonstrate that LinkBlackHole and LinkBlackHole* are more robust than the state-of-the-art algorithms when the networks become increasingly mixed and/or overlapping in the communities.

5.2 Community Quality for Real-World Networks

5.2.1 Data sets

The data sets are real-world networks obtained from the Stanford Large Network Data Set Collection [29]. They comprise 14 data sets that collectively cover the following eight categories: social networks, collaboration networks, citation networks, communication networks, product networks, road networks, autonomous system networks, and peer-to-peer networks. Table 3 summarizes the network profiles of the data sets. For each network, its average degree of a node $\langle k \rangle$ was measured and used to calculate the belonging coefficient cutoff threshold τ (see Section 4.4).

TABLE 3: Profiles of real-world networks.

Category	Data set	Number of nodes	Number of edges	Ave. deg.
Social	Brightkite	58,228	214,078	17.48
	Epinions	75,879	508,837	13.41
	Slashdot	77,360	905,468	23.41
Collaboration	col-HepPh	12,008	118,521	19.74
	AstroPh	18,772	198,110	21.11
	CondMat	23,133	93,497	8.08
	DBLP	317,080	1,049,866	6.62
Citation	cit-HepTh	27,770	352,807	25.41
Communication	Enron-email	36,692	183,831	10.02
Product	Amazon	334,863	925,872	5.53
Road	RoadNet-PA	1,088,092	1,541,898	2.83
Autonomous system	Oregon	10,900	31,180	5.72
	Caida	26,475	106,762	8.07
Peer-to-peer	Gnutella	26,518	65,369	4.93

5.2.2 Performance Measure

There is no ground truth available for the real-world networks and, in such a case, *overlapping modularity* M^{ov} is a suitable, popular cluster quality measure of overlapping communities [44]. For each cluster, its overlapping modularity measures how densely modularized (i.e., connected inward and disconnected outward) the connected nodes are in the cluster and is calculated by aggregating over all nodes in the cluster the differences between the number of incoming links and the number of outgoing links relative to the degree of each node. The aggregate value is normalized to the range of -1.0 to 1.0, where 1.0 means the maximum modularity.

While the modularity is an effective quality measure, it tends to penalize small communities because they may not have enough nodes to be aggregated over. Thus, *clustering coverage* CC was used as another counter-balancing measure. Its value is calculated as the fraction of nodes having a cluster membership [1] and, thus, gives equal merit to all nodes regardless of the size of the cluster they belong to. The coverage value ranges from 0.0 to 1.0, where 1.0 means that *every* node belongs to a certain cluster.

5.2.3 Results

Figure 13 shows the results for the real-world networks. M^{ov} and CC, each normalized to the range of 0 to 1, are overlaid in the same plot so that their values can be compared between algorithms not only as a whole measure but also in separate measures. Trivial clusters with fewer than three nodes have been excluded from consideration when calculating M^{ov} and CC. Each individual algorithm has its own sets of performance parameters, and they have been

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2018.2873750, IEEE Transactions on Knowledge and Data Engineering

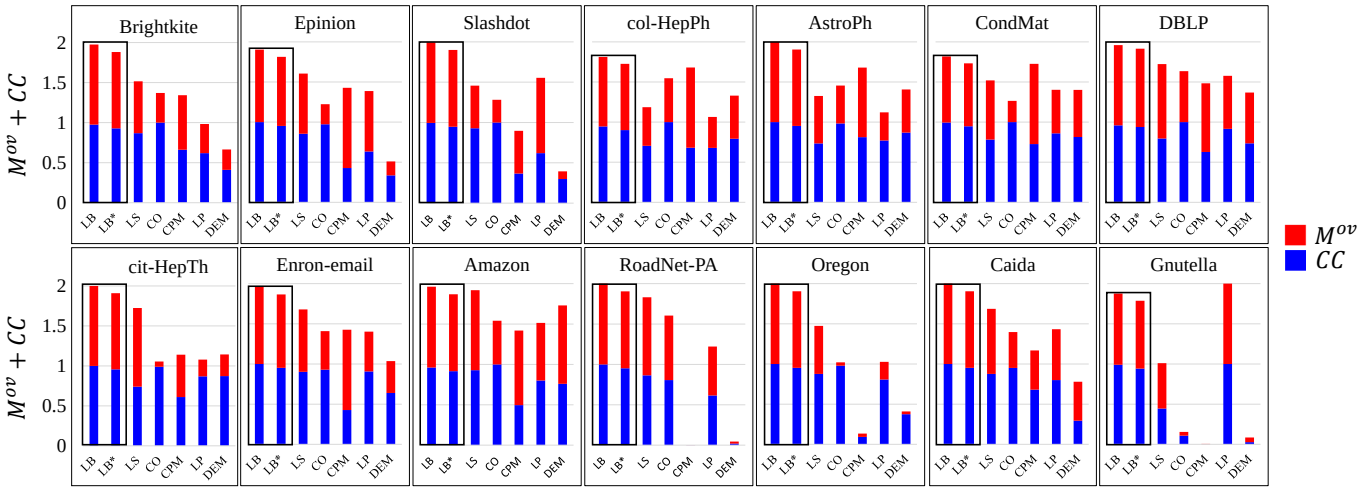
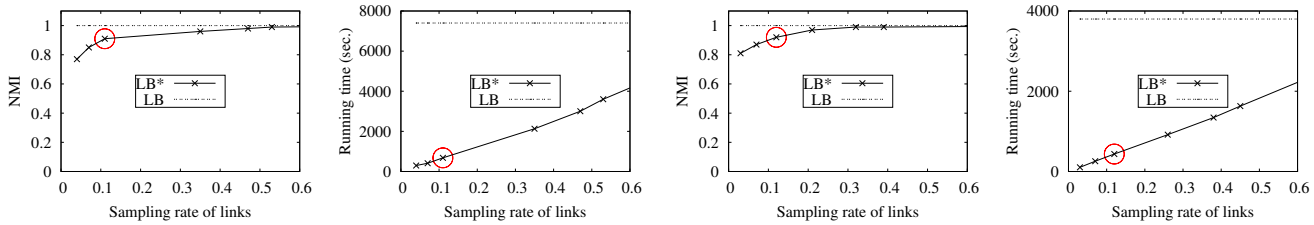


Fig. 13: Results for real-world networks.



(a) Accuracy (Brightkite). (b) Running time (Brightkite). (c) Accuracy (col-HepPh). (d) Running time (col-HepPh).

Fig. 14: Effect of sampling rate on LinkBlackHole* performance.

tuned respectively to achieve the best performance. The results are impressive. LinkBlackHole* and LinkBlackHole show significantly better performance than the other algorithms for 13 out of 14 data sets. Even in the one exception for Gnutella, LinkPartition shows only *slightly* (6%) better performance than LinkBlackHole* and LinkBlackHole, which are the close second best performers.

Among the other algorithms, LinkSCAN* generally shows a decent, next best performance to LinkBlackHole* and LinkBlackHole; COPRA shows high clustering coverage but somewhat low modularity because it is a partitioning-based algorithm and, therefore, assigns community memberships to every node; CPM shows relatively decent performance in dense networks, but very poor performance in sparse networks (e.g., RoadNet-PA ($\langle k \rangle = 2.83$), Oregon ($\langle k \rangle = 5.72$), and Gnutella ($\langle k \rangle = 4.93$)) because it is hard to detect the base structure (i.e., clique) in sparse networks; LinkPartition and DEMON consistently perform poorly on most real-world data sets because LinkPartition tends to discover many small communities while DEMON tends to discover very large communities compared with the communities discovered by other algorithms.

5.3 Effects of Sampling on LinkBlackHole*

The clustering accuracy and running time of LinkBlackHole* were compared with those LinkBlackHole while varying the link sampling rate in the networks. The sampling rate was controlled by doubling the value of α from $\frac{1}{8}\langle k \rangle$ to $8\langle k \rangle$ while fixing β to 1.0, where $\langle k \rangle$ is the average degree of a node in the original graph.

Figure 14 shows the results for Brightkite and col-HepPh network data sets. These two data sets represent each of the two main cohorts in Table 3, and the other data sets also showed similar results. Naturally, the community accuracy and running time of LinkBlackHole* approach those of LinkBlackHole as the sampling rate increases. Even when the sampling rate is only 10% (marked by a red circle in the figure), which corresponds to $\alpha = \frac{1}{2}\langle k \rangle$ and $\beta = 1$ according to the empirical result mentioned in Section 4.1.2, the accuracy of LinkBlackHole* still exceeds 90% that of LinkBlackHole while the running time is reduced by approximately 10 times. This solid result demonstrates LinkBlackHole*'s strong resilience to sampling, that it significantly improves the efficiency of LinkBlackHole with negligible compromise of the accuracy.

5.4 Running Time Scalability

Because the algorithms compared are implemented in different languages (e.g., Java, C/C++, Matlab, and Python), the primary intention of this test is to examine the running time increase of *each* algorithm, including LinkBlackHole*, as the data size increases. Scalability was measured particularly over network density by increasing the number of edges in the network from 100,000 to 3,200,000 in the synthetic LFR benchmark data sets. Scalability with the network density is meaningful because what LinkBlackHole* does is *link* clustering.

Figure 15 shows the result. It shows that LinkBlackHole* achieves near-linear scalability, and so do all the other algorithms except CPM and DEMON, which show faster, apparently exponential growth of running time. As the number of links doubled from $10 \cdot 10^4$ to $320 \cdot 10^4$ five times, the running time of LinkBlackHole* increased by 2.1, 2.2, 2.2, 2.6, and 2.5 times, but that of DEMON increased by 2.4,

2.6, 3.9, 5.6, and 4.7 times. Although LinkBlackHole* takes longer than the other algorithms, it is still sufficiently fast, finishing in approximately 2 hours on a single machine for dense networks with over 1 million edges.

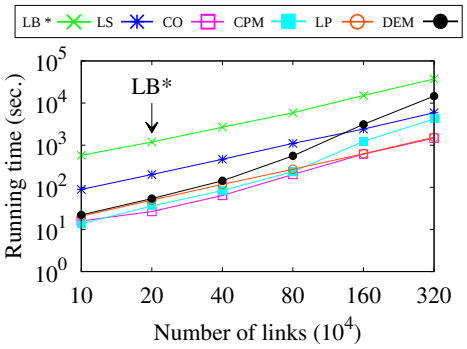


Fig. 15: Running time scalability with the number of links.

6 Conclusion

LinkBlackHole* combines LinkSCAN*'s ability to detect overlapping communities and BlackHole's robustness to mixing between communities, and thereby can accurately detect highly-mixed overlapping communities from large networks. Specifically, it combines the link-space transformation of LinkSCAN* and the black hole transformation of BlackHole to materialize link embedding.

Extensive evaluation through synthetic network experiments and real-world network experiments demonstrated the performance merit of LinkBlackHole and its link-sampled version LinkBlackHole* compared with other algorithms that define the state of overlapping community detection. LinkBlackHole* and LinkBlackHole showed the best quality of mixed overlapping communities as reflected in the clustering accuracy, and, more notably, the accuracy advantage increased as the degree of community mixing and/or overlap increased. In addition, LinkBlackHole* showed very strong resilience to the link sampling and also showed acceptable running time scalability with the number of links.

As future work, we plan to develop a *parallel* embedding algorithm that can simultaneously run on multiple machines to further improve the running time of LinkBlackHole* as well as BlackHole.

Acknowledgment

This work was supported by the MOLIT (The Ministry of Land, Infrastructure and Transport), Korea, under the national spatial information research program supervised by the KAIA (Korea Agency for Infrastructure Technology Advancement) (18NSIP-B081011-05).

References

[1] Y.-Y. Ahn, J. P. Bagrow, and S. Lehmann. Link communities reveal multiscale complexity in networks. *Nature*, 466(7307):761–764, 2010.

[2] A. Banerjee and R. N Dave. Validating clusters using the hopkins statistic. In *Proc. 2004 IEEE Int'l Conf. on Fuzzy systems*, pages 149–153, 2004.

[3] D. Chen, M. Shang, Z. Lv, and Y. Fu. Detecting overlapping communities of weighted networks via a local algorithm. *Physica A: Statistical Mechanics and its Applications*, 389(19):4177–4187, 2010.

[4] W. Chen, Z. Liu, X. Sun, and Y. Wang. A game-theoretic framework to identify overlapping communities in social networks. *Data Mining and Knowledge Discovery*, 21(2):224–240, 2010.

[5] H. Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *The Annals of Mathematical Statistics*, pages 493–507, 1952.

[6] M. Coscia, G. Rossetti, F. Giannotti, and D. Pedreschi. Uncovering hierarchical and overlapping communities with a local-first approach. *ACM Trans. on Knowledge Discovery from Data*, 9(1):6, 2014.

[7] L. Danon, A. Diaz-Guilera, J. Duch, and A. Arenas. Comparing community structure identification. *Journal of Statistical Mechanics: Theory and Experiment*, 2005(9):P09008, 2005.

[8] R. Davidson and D. Harel. Drawing graphs nicely using simulated annealing. *ACM Trans. on Graphics*, 15(4):301–331, 1996.

[9] A. Epasto, S. Lattanzi, and R. Paes Leme. Ego-splitting framework: From non-overlapping to overlapping clusters. In *Proc. 23rd ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining*, pages 145–154, 2017.

[10] M. Ester, H.-P. Kriegel, J. Sander, M. Wimmer, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proc. 2nd ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining*, pages 291–316, 1996.

[11] T. S. Evans and R. Lambiotte. Line graphs, link partitions, and overlapping communities. *Physical Review E*, 80(1):016105, 2009.

[12] T. M. J. Fruchterman and E. M. Reingold. Graph drawing by force-directed placement. *Software - Practice and Experience*, 21(11):1129–1164, 1991.

[13] Q. Fu and A. Banerjee. Multiplicative mixture models for overlapping clustering. In *Proc. 2008 IEEE Int'l Conf. on Data Mining*, pages 791–796, December 2008.

[14] S. Gregory. An algorithm to find overlapping community structure in networks. In *Proc. 11th European Conf. on Principles and Practice of Knowledge Discovery in Databases*, pages 91–102, 2007.

[15] S. Gregory. Finding overlapping communities in networks by label propagation. *New Journal of Physics*, 12:103018, 2010.

[16] J. Han, M. Kamber, and J. Pei. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2011.

[17] L. Šubelj and M. Bajec. Unfolding communities in large complex networks: Combining defensive and offensive label propagation for core extraction. *Physical Review E*, 83(3):036103, 2011.

[18] M. Jacomy, T. Venturini, S. Heumann, and M. Bastian. ForceAtlas2, a continuous graph layout algorithm for handy network visualization designed for the gephi software. *PLOS ONE*, 9(6):e98679, 2014.

[19] J. Kim, J.-G. Lee, and S. Lim. Differential flattening: A novel framework for community detection in multi-layer graphs. *ACM Trans. on Intelligent Systems and Technology*, 8(2):27, 2017.

[20] Y. Kim and H. Jeong. Map equation for link communities. *Physical Review E*, 84:026110, 2011.

[21] I. A. Kovàcs, R. Palotai, M. S. Szalay, and P. Csermely. Community landscapes: An integrative approach to determine overlapping network module hierarchy, identify key nodes and predict network dynamics. *PLoS ONE*, 5(9):e12528, 2010.

[22] J. M. Kumpula, M. Kivelä, K. Kaski, and J. Saramäki. Sequential algorithm for fast clique percolation. *Physical Review E*, 78(2):026109, 2008.

[23] A. Lancichinetti and S. Fortunato. Community detection algorithms: A comparative analysis. *Physical Review E*, 80:056117, 2009.

[24] A. Lancichinetti, S. Fortunato, and J. Kertész. Detecting the overlapping and hierarchical community structure in complex networks. *New Journal of Physics*, 11:033015, 2009.

[25] A. Lancichinetti, F. Radicchi, J. J. Ramasco, and S. Fortunato. Finding statistically significant communities in networks. *PLoS ONE*, 6(4):e18961, 2011.

[26] C. Lee, F. Reid, A. F. McDaid, and N. J. Hurley. Detecting highly overlapping community structure by greedy clique expansion. In *Proc. 4th Workshop on Social Network Mining and Analysis*, 2015.

[27] J.-D. Leeuw. Convergence of the majorization method for multidimensional scaling. *Journal of Classification*, 5(2):163–180, 1988.

[28] J. Leskovec and C. Faloutsos. Sampling from large graphs. In *Proc. 12th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining*, pages 631–636, 2006.

[29] J. Leskovec and A. Krevl. SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data, jun 2014.

[30] Y. Li, K. He, D. Bindel, and J.-E Hopcroft. Uncovering the small community structure in large networks: A local spectral approach. In *Proc. 24th Int'l Conf. on World Wide Web*, pages 658–668, 2015.
[31] D. Liben-Nowell and J. Kleinberg. The link-prediction problem for social networks. *Journal of the American Society for Information Science and Technology*, 58(7):1019–1031, 2007.
[32] S. Lim, J. Kim, and J.-G. Lee. BlackHole: Robust community detection inspired by graph drawing. In *Proc. 32nd IEEE Int'l Conf. on Data Engineering*, pages 25–36, 2016.
[33] S. Lim, S. Ryu, S. Kwon, K. Jung, and J.-G. Lee. LinkSCAN*: Overlapping community detection using the link-space transformation. In *Proc. 30th IEEE Int'l Conf. on Data Engineering*, pages 292–303, 2014.
[34] T. Nepusz, A. Petróczi, L. Négyessy, and F. Bazsó. Fuzzy communities and the concept of bridgeness in complex networks. *Physical Review E*, 77(1):016107, 2008.
[35] A. Noack. Energy models for graph clustering. *Journal of Graph Algorithms and Applications*, 11(2):453–480, 2007.
[36] A. Noack. Modularity clustering is force-directed layout. *Physical Review E*, 79(2):026102, 2009.
[37] G. Palla, I. Derényi, I. Farkas, and T. Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435(7043):814–818, 2005.
[38] S. V. Pemmaraju and C. A. Shaffer. Analysis of the worst case space complexity of a PR quadtree. *Information Processing Letters*, 49(5):263–267, 1994.
[39] I. Psorakis, S. Roberts, M. Ebden, and B. Sheldon. Overlapping community detection using bayesian non-negative matrix factorization. *Physical Review E*, 83(6):066114, 2011.
[40] B. S. Rees and K. B. Gallagher. Overlapping community detection by collective friendship group inference. In *Proc. 2010 Int'l Conf. on Advances in Social Networks Analysis and Mining*, pages 375–379, 2010.
[41] Wei Ren, Guiying Yan, and Xiaoping Liao. Simple probabilistic algorithm for detecting community structure. *Physical Review E*, 79(3):036111, 2009.
[42] J. K. Salmon. *Parallel Hierarchical N-body Methods*. PhD thesis, California Institute of Technology, 1991.
[43] H. Shen, X. Cheng, K. Cai, and M.-B. Hu. Detect overlapping and hierarchical community structure in networks. *Physica A: Statistical Mechanics and its Applications*, 388(8):1706–1712, 2009.
[44] H.-W. Shen, X.-Q. Cheng, and J.-F. Guo. Quantifying and identifying the overlapping community structure in networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2009(7):P07042, 2009.
[45] R. Tamassia. *Handbook of Graph Drawing and Visualization*. Chapman and Hall/CRC, 2013.
[46] S. Tang, J. Yuan, X. Mao, X.-Y. Li, W. Chen, and G. Dai. Relationship classification in large scale osn and its impact on information propagation. In *Proc. 30th IEEE Int'l Conf. on Computer Communications (INFOCOM)*, pages 1661–1669, 2011.
[47] J. J. Whang, D. F. Gleich, and I. S. Dhillon. Overlapping community detection using neighborhood-inflated seed expansion. *IEEE Trans. on Knowledge and Data Engineering*, 28(5):1272–1284, 2016.
[48] J. Xie, S. Kelley, and B. K. Szymanski. Overlapping community detection in networks: the state of the art and comparative study. *ACM Computing Surveys*, 45(4):43, 2013.
[49] J. Xie, B. K. Szymanski, and X. Liu. Slpa: Uncovering overlapping communities in social networks via a speaker-listener interaction dynamic process. In *Proc. IEEE 11th Int'l Conf. on Data Mining Workshops*, pages 344–349, 2011.
[50] B. Yan and S. Gregory. Detecting communities in networks by merging cliques. In *Proc. 2009 IEEE Int'l Conf. on Intelligent Computing and Intelligent Systems*, pages 832–836, 2009.
[51] S. Zhang, R.-S. Wang, and X.-S. Zhang. Identification of overlapping community structure in complex networks using fuzzy c-means clustering. *Physica A: Statistical Mechanics and its Applications*, 374(1):483–490, 2007.
[52] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: An efficient data clustering method for very large databases. In *Proc. 1996 ACM SIGMOD Int'l Conf. on Management of Data*, pages 103–114, 1996.

Jungeun Kim is a postdoctoral researcher at the Graduate School of Knowledge Service Engineering, KAIST. He received BS (2011) in Information and Communication Engineering from Inha University and MS (2013) and PhD (2017) in Knowledge Service Engineering from KAIST. His research interests include social-network and graph big data mining, big data analysis with the distributed processing platforms, and stream data mining.

Sungsu Lim is an Assistant Professor of Computer Science and Engineering at Chungnam National University. He received BS (2009) and MS (2011) in Mathematical Sciences from KAIST and PhD (2016) in Knowledge Service Engineering from KAIST. His research interests include mining and modeling large-scale social and information networks and their theoretical aspects.

Jae-Gil Lee is an Associate Professor at KAIST and is leading the Data Mining Lab. Previously, he was a postdoctoral researcher at IBM Almaden Research Center and a postdoc research associate at The University of Illinois Urbana-Champaign. He received BS, MS, and PhD in Computer Science from KAIST. His research interests include spatio-temporal data mining, social-network and graph data mining, and big data analysis with Hadoop/MapReduce.

Byung Suk Lee is a Professor of Computer Science at The University of Vermont. He received BS in Electronics Engineering from Seoul National University, MS in Electrical Engineering from KAIST, and PhD in Electrical Engineering from Stanford University. His research interests include database, data mining, data stream with a focus on models, algorithms, and performance.