LinkSCAN*: Overlapping Community Detection Using the Link-Space Transformation

Sungsu Lim [†1], Seungwoo Ryu [‡2], Sejeong Kwon [§3], Kyomin Jung [¶4], Jae-Gil Lee [†5*]

[†] *Department of Knowledge Service Engineering, KAIST*
[‡] *Samsung Advanced Institute of Technology, Samsung Electronics*
[§] *Graduate School of Culture Technology, KAIST*
[¶] *Department of Electrical and Computer Engineering, Seoul National University*
[1,3,5] {ssungssu, gsj1029, jaegil}@kaist.ac.kr, [2] seungwoo.ryu@samsung.com, [4] kjung@snu.ac.kr

***Abstract*—In this paper, for overlapping community detection, we propose a novel framework of the *link-space transformation* that transforms a given original graph into a *link-space graph*. Its unique idea is to consider topological structure and link similarity separately using two distinct types of graphs: the line graph and the original graph. For topological structure, each link of the original graph is mapped to a node of the link-space graph, which enables us to discover *overlapping* communities using *non*-overlapping community detection algorithms as in the line graph. For link similarity, it is calculated on the original graph and carried over into the link-space graph, which enables us to keep the original structure on the transformed graph. Thus, our transformation, by combining these two advantages, facilitates overlapping community detection as well as improves the resulting quality. Based on this framework, we develop the algorithm *LinkSCAN* that performs structural clustering on the link-space graph. Moreover, we propose the algorithm *LinkSCAN** that enhances the efficiency of LinkSCAN by *sampling*. Extensive experiments were conducted using the LFR benchmark networks as well as some real-world networks. The results show that our algorithms achieve higher accuracy, quality, and coverage than the state-of-the-art algorithms.**

I. Introduction

A. Motivation

In many real-world social networks such as Facebook and Twitter, individuals can belong to multiple communities, e.g., family, friends, colleagues, and schoolmates [1]. Thus, there have been active discussions on *overlapping* community detection [2], [3], [4], [5], [6], [1]. The existing methods can be roughly classified into two categories depending on the graph element used for community discovery.

- **Node-based**: Each node is directly associated with multiple communities. A popular method is labeling a node x with a set of pairs (c, b), where c is a community identifier and b is a belonging coefficient [6]. A *belonging coefficient* indicates the strength of x's membership of the community c. Well-known methods in this category include the label propagation method [6].
- **Structure-based**: Community discovery is done through a pre-defined structure such as a clique or a link. A node can participate in *multiple* cliques or links. Thus, even if cliques or links are partitioned into disjoint communities, participating nodes can belong to multiple communities. Well-known methods in this category include the Clique Percolation Method (CPM) [3], [1] and the link-partition method [2], [4].

In connection with this categorization, we have found out that the existing methods suffer from three common problems.

1) *Many highly overlapping nodes*: The node-based category assigns a set of belonging coefficients into a node, where the sum of the coefficients is 1. Thus, as more communities overlap at a node, the value 1 needs to be distributed to more communities, resulting in smaller differences among the coefficients. If the coefficients get close with each other, it would be tricky to distinguish the communities to which the node belongs and those to which the node does not.
2) *Incorrect base-structures*: The structure-based category first tries to discover base-structures from a graph. This procedure is based on the assumption that the graph consists of many such base-structures. However, in the CPM, the graph may not have many cliques since cliques are very strict structures. Thus, for loosely-connected graphs, many of the nodes are not covered by any community in the CPM. The link-partition method does not suffer from this problem since the link is the basic structure of a graph.
3) *Incorrect membership of weak ties*: In the structure-based category, it is usually required that *every* base-structure should belong to at least one community. This requirement is problematic since it may result in *overly*-overlapping communities. For example, a weak tie [7] had better *not* be included in any community since it does not represent strong relationship between the nodes. In Figure 1, suppose that Jack and Bob are travel buddies and do not have common friends in a social network. Jack belongs to his workplace community, and Bob to his family community. If the weak tie is assigned to one of the communities, the two communities overly overlap at either Jack or Bob. In fact, it is obvious that the two communities should be separated.

Table I compares the popular community detection algorithms for these three problems. An 'X' mark indicates that

* Jae-Gil Lee is the corresponding author.

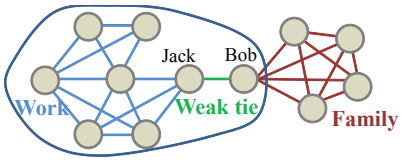


Fig. 1. An example of a weak tie and its improper membership.

the algorithm does *not* suffer from the problem. Our algorithm is designed to be free from the three problems.

TABLE I
COMPARISON OF THE ALGORITHMS.

	Label Propagation	CPM	Link Partition	LinkSCAN* (ours)
Problem 1	O	X	X	**X**
Problem 2	X	O	X	**X**
Problem 3	X	O	O	**X**

B. Our Solution

In this paper, with considering these drawbacks, we propose a new framework, which we call the *link-space transformation*. This transformation converts an original graph to a *link-space* graph. Each node of the link-space graph is a link of the original graph, and the nodes of the link-space graph are connected if the corresponding links in the original graph are connected through some node. In the topological point of view, the link-space graph is identical to the line graph proposed by Evans and Lambiotte [4]. The difference is how a weight is assigned to each link: the weight is calculated *on the transformed graph* (i.e., the line graph) in the previous approach whereas it is calculated *on the original graph* in our approach. That is, the similarity on the original graph is carried over into the link-space graph in our approach.

We contend that the link-space graph combines the advantages of the original graph and the line graph as shown in Figure 2. Inheriting the advantages of the line graph, finding *disjoint* communities enables us to find *overlapping* communities. Inheriting the advantages of the original graph, its original structure is better preserved in the link-space graph than in the line graph.

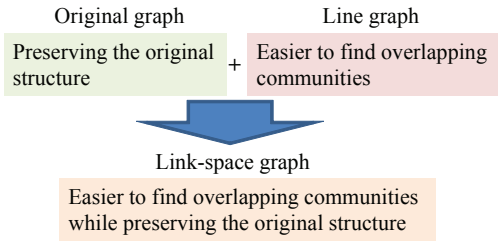


Fig. 2. The advantages of the link-space transformation.

Example *1:* Figure 3 contrasts the link-space graph with the line graph. For now, please look at the relative size of the similarity scores. In Figure 3(a), let's consider the similarity between e_{12} and e_{13} as well as that between e_{13} and e_{23}. It will be natural if these two similarity scores are the same because the pair of links share one endpoint and are connected by another link. In Figure 3(b), the latter is smaller than the former because the common node 3 of e_{13} and e_{23} has a higher degree than the common node 1 of e_{12} and e_{13}. A critical problem of the line graph is that the weight heavily depends on the degree of a node in the original graph. On the other hand, in Figure 3(c) of the link-space graph, the similarity scores properly reflect the structure of the original graph. □

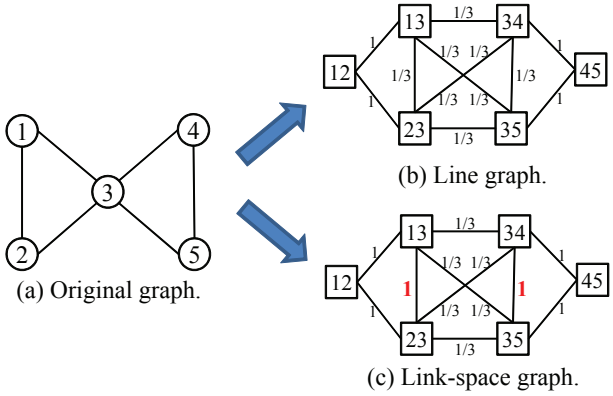


Fig. 3. Contrast between the line graph and the link-space graph.

We now explain how the link-space transformation can solve the problems listed in Table I.

1) *Solution to Problem 1*: Finding *disjoint* communities of the nodes in the link-space graph obviates the need for a belonging coefficient.
2) *Solution to Problem 2*: A *link* is used as the base-structure since a node of the link-space graph corresponds to a link of the original graph.
3) *Solution to Problem 3*: We introduce the notion of *membership neutrality* that allows a link to be neutral to any community. Since any *non*-overlapping community detection algorithm can be adopted owing to the link-space transformation, it is very easy to find an algorithm satisfying membership neutrality.

In this paper, based on the link-space transformation, we develop an overlapping community detection algorithm, which we call *LinkSCAN (Link Structural Clustering Algorithm for Networks)*. Then, we develop an enhanced algorithm *LinkSCAN** that improves the efficiency of LinkSCAN by sampling of links. Since the number of links significantly increases in the link-space graph compared with the original graph, reducing the links considered helps improve the efficiency of the algorithm.

Our approach of using the link-space transformation is not dependent on a specific community detection algorithm. However, as discussed above, the algorithm should satisfy *membership neutrality*. We decide to adopt the algorithm SCAN [8] since it identifies the hub or outlier nodes that do not belong to any community.

Our proposed algorithms take a graph $\mathcal{G} = (V, E)$ as an input, where V and E represent the set of all nodes (individuals) and the set of all links (relationships) respectively, and produce an overlapping clustering C. We define an overlapping clustering by a collection of sets of nodes where each set consists of the nodes having a specific type of relationship with another node in the same set. LinkSCAN* consists of four major steps. First, the original graph $\mathcal{G}$ is converted to a link-space graph $LS(\mathcal{G})$ by the link-space transformation. Second, the links of $LS(\mathcal{G})$ are sampled. Third, a variation of the algorithm SCAN is applied to $LS(\mathcal{G})$, and the disjoint communities of the nodes are obtained. Fourth, the membership on $LS(\mathcal{G})$ is translated back to that on $\mathcal{G}$. The only difference between LinkSCAN and LinkSCAN* is the existence of the second step.

Example *2:* Figure 4 shows the four steps of LinkSCAN* for a simple graph. The grey links of the link-space graph that are *not* sampled are ignored during clustering. As a result of clustering, the nodes in the link-space graph are partitioned into two communities in red and in blue. The membership of a link determines that of its endpoint nodes. The set of nodes $\{1, 2, 3, 4\}$ belong to the red community, and that of nodes $\{4, 5, 6, 7, 8\}$ to the blue community. Thus, the two communities overlap at the node 4. □

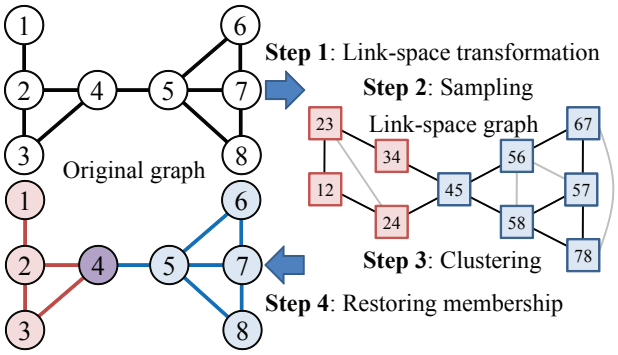


Fig. 4. The overall procedure of overlapping community detection.

C. Key Contributions

In summary, the contributions of this paper are as follows.

- We identify three common drawbacks of the previous algorithms for overlapping community detection by analyzing large-scale real-world social networks.
- We propose a novel framework called the *link-space transformation* that incorporates the advantages of both the original graph and the line graph.
- We introduce the notion of *membership neutrality*, which is essential in analyzing complicated, real-world networks.
- We develop the algorithm *LinkSCAN* satisfying membership neutrality in the framework of the link-space transformation.
- We present the enhanced algorithm *LinkSCAN** that achieves higher efficiency with negligible errors compared with LinkSCAN.

The rest of the paper is organized as follows. Section II explains the background knowledge for our work. Section III proposes the link-space transformation and develops the algorithms LinkSCAN and LinkSCAN*. Section IV presents the results of evaluation. Section V summarizes state-of-the-art related work. Finally, Section VI concludes this study.

II. Preliminaries

Our goal is to identify the overlapping community structure of a given network. In this section, we review (i) link clustering that has been widely used for this purpose and (ii) structural clustering which is adopted in our algorithm.

A. Link Clustering

Recently, some researchers have focused on the community detection based on link clustering [2], [4], [9]. The community of links is preferable to that of nodes from a theoretical point of view, because the link is likely to have a unique identity whereas the node tends to have multiple identities [10]. Link clustering is formally defined in Definition 1.

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

Most notably, Ahn et al. [2] proposed the link-partition method using a Jaccard-type similarity score between links. The membership of a node is determined by those of its incident links. Please note that a node participates in multiple communities if the node has multiple incident links with different types of relationship.

In fact, 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 $\mathcal{G}$. To construct $L(\mathcal{G})$, each link of $\mathcal{G}$ becomes 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 $\mathcal{G}$. However, $L(\mathcal{G})$ gives too much prominence to the high-degree nodes of the original graph $\mathcal{G}$ since a node of degree d_i in $\mathcal{G}$ is mapped to a clique of size d_i in $L(\mathcal{G})$, which has $d_i(d_i - 1)/2$ links [4].

B. Structural Clustering

The algorithm SCAN [8] has been regarded as one of the most popular structural clustering algorithms for networks. It uses the structural similarity s for every two nodes v and w, which is defined by Eq. (1). Here, $\Gamma(v) = \{w \in V | (v, w) \in E\} \cup \{v\}$. A node v is said to be a *core* node with respect to ϵ and μ if $|N_\epsilon(v)| \geq \mu$, where $N_\epsilon(v)$ is the set of all nodes that have at least ϵ similarity with v. Hence, a node becomes a core node if and only if it is well-connected to other nodes.

$$s(v, w) = \frac{|\Gamma(v) \cap \Gamma(w)|}{\sqrt{|\Gamma(v)||\Gamma(w)|}} \tag{1}$$

Let us explain the basic definitions for SCAN. For any node $u \in N_\epsilon(v)$ for some core node v, a node u is said to be *directly reachable* from v. Furthermore, a node u is *reachable* from v if there is a sequence of nodes $\{v_0, v_1, \ldots, v_t\}$ for

some $t \in \mathbb{N}$ such that $v_0 = v$, $v_t = u$, and v_{k+1} is directly reachable from v_k for all $0 \leq k \leq t-1$. Remark that the reachability is a symmetric relation for a pair of core nodes, but it is not otherwise. A pair of nodes u and v are said to be *structure-connected* if there is a node w such that both u and v are reachable from w. Then, the connectivity becomes a symmetric relation for any pair of nodes. Finally, a *cluster* is defined as a maximal set of structured-connected nodes.

Non-core nodes may not participate in any cluster. When a node v is not a member of any cluster, v is called a *hub* node if it bridges at least two communities in the network or is called an *outlier* node otherwise.

III. Our Proposed Framework

Our proposed algorithm *LinkSCAN** consists of the four major steps shown in Algorithm 1. The baseline algorithm *LinkSCAN* omits the sampling step, and the other steps are exactly the same in both algorithms. That is, LinkSCAN* considers only a small (typically, less than 30%) proportion of links in the link-space graph whereas LinkSCAN all the links. LinkSCAN* improves efficiency by at least 200% with introducing less than 10% errors compared with LinkSCAN.

Algorithm 1 LinkSCAN and **LinkSCAN***

INPUT: (i) a graph $\mathcal{G} = (V, E)$
(ii) parameters ϵ and μ for structural clustering
(iii) parameters α and β for link sampling
OUTPUT: an overlapping clustering NC of $\mathcal{G}$;

```
/* Step 1. Link-Space Transformation (Algorithm 2) */
LS(G) = (V', E') ← LinkSpaceTransformation(G);
/* Step 2. Link Sampling */
if LinkSCAN* then
    /* Only LinkSCAN* requires this step. */
    LS(G) ← LinkSampling(LS(G), α, β);
end if
/* Step 3. Structural Clustering (Algorithm 3) */
/* LC = {V_C' | V_C' ⊆ V'} */
LC ← StructuralClustering(LS(G), ϵ, μ);
/* Step 4. Membership Translation */
/* NC = {V_C | V_C ⊆ V} */
NC ← MembershipTranslation(LC, LS(G));
return NC
```

In the remaining of this section, we explain the main ingredients: link-space transformation (Section III-A), membership neutrality (Section III-B), and structural clustering (Section III-C). For ease of exposition, we present first LinkSCAN (Section III-D) and then LinkSCAN* (Section III-E). Before proceeding to algorithm descriptions, in Table II, we summarize the notation used throughout this paper.

A. Link-Space Transformation

1) Formal Definition: The beauty of the link-space transformation is to consider the two types of graphs—the original graph and the line graph—*with taking their advantages only*. That is, it allows us to find *overlapping* communities by using *non*-overlapping community detection algorithms and, at the same time, to respect the structure of the original graph. This idea is developed roughly based on the kernel trick [11]. In the kernel trick, the support vector machine (SVM) attempts to find a separating plane in a *higher-dimensional* feature space, but the kernel function is evaluated for the *original* data points. In this sense, the link-space transformation resembles the kernel trick.

TABLE II
SUMMARY OF THE NOTATION.

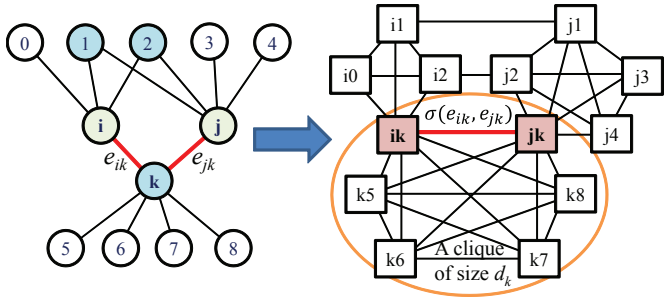
Notation	Description
$\mathcal{G}$	an original graph
$LS(\mathcal{G})$	the link-space graph of $\mathcal{G}$
v_i	a node in an original graph
e_{ij} or $e_{(i,j)}$	a link in an original graph
$v_{e_{ij}}$ or $v_{e_{(i,j)}}$	a node in a link-space graph
$\sigma(\cdot,\cdot)$	similarity between two links in $\mathcal{G}$
$\omega(\cdot,\cdot)$	similarity between two nodes in $LS(\mathcal{G})$
d_v	the degree of a node v
n_v	the number of links sampled at a node v

The *link-space transformation* is formally defined in Definition 2. The difference between the link-space graph and the line graph is only the third item on how the similarity between links is derived.

Definition *2:* Given a graph $\mathcal{G}$, its *link-space graph* $LS(\mathcal{G})$ is a graph such that

- A node $v_{e_{ij}}$ of $LS(\mathcal{G})$ represents the link e_{ij} between the nodes v_i and v_j of $\mathcal{G}$;
- Two nodes $v_{e_{ik}}$ and $v_{e_{jk}}$ of $LS(\mathcal{G})$ are adjacent if and only if their corresponding links share a common endpoint in $\mathcal{G}$; and
- 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 $\mathcal{G}$.

Example *3:* Figure 5(b) is the link-space graph transformed from Figure 5(a). The two colored nodes $v_{e_{ik}}$ and $v_{e_{jk}}$ in $LS(\mathcal{G})$ correspond to the two links e_{ik} and e_{jk} in $\mathcal{G}$, respectively. The weight of the link between $v_{e_{ik}}$ and $v_{e_{jk}}$ is calculated using e_{ik} and e_{jk} on $\mathcal{G}$. □



(a) Original graph $\mathcal{G}$. (b) Link-space graph $LS(\mathcal{G})$.

Fig. 5. An example of the link-space transformation.

One might argue that it would be better to simply calculate the similarity between the nodes on the link-space graph. However, the node similarity on the link-space graph could be distorted since it highly depends on the degree of the common

endpoint of the original graph. For example, the structural similarity of Eq. (1), which ranges from 0 to 1, is $\frac{6}{\sqrt{9 \cdot 10}} = 0.63$ for the nodes $v_{e_{ik}}$ and $v_{e_{jk}}$ in Figure 5(b). This similarity score tends to be too high since it is elevated too much owing to a high degree of the common node v_k.

2) Link Similarity: We adopt the link similarity proposed by Ahn et al. [2] for $\sigma(\cdot,\cdot)$ of Definition 2. This measure is chosen because it is shown to be very effective despite its simplicity. Note that any other similarity measure, e.g., one of the measures in [12], can be used.

Our link similarity $\sigma(\cdot,\cdot)$ is defined as Eq. (2). Again, $\Gamma(v) = \{w \in V | (v,w) \in E\} \cup \{v\}$, and $d(v,u)$ is the length of the shortest path between v and u. Intuitively, our approach gives a more natural way to compute the similarity between two incident links, because it is free from the degree of the common endpoint v_k.

$$\omega(v_{e_{ik}}, v_{e_{jk}}) = \sigma(e_{ik}, e_{jk}) = \frac{|\Gamma(v_i) \cap \Gamma(v_j)|}{|\Gamma(v_i) \cup \Gamma(v_j)|} \qquad (2)$$

Example *4:* In Figure 5(a), $\Gamma(v_i) = \{v_0, v_1, v_2, v_i, v_k\}$ 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 $\Gamma(v_i) \cup \Gamma(v_j) = \{v_0, v_1, v_2, v_3, v_4, v_i, v_j, v_k\}$. Finally, the similarity score is $\sigma(e_{ik}, e_{jk}) = \frac{3}{8} = 0.375$, which looks reasonable. □

B. Membership Neutrality

One major drawback of link clustering is that it produces too many nodes that join multiple communities, because a node participates in a cluster even if there is only one incident link having the relationship corresponding to the cluster. To resolve this problem, our main idea is to relax the definition of link clustering in Definition 3.

Definition *3:* Let $\mathcal{G} = (V, E)$ be a given network. Then, a collection P'' of subsets of E is said to be a *generalized link clustering* of $\mathcal{G}$ if each link participates in at most one link cluster of P''.

That is, we allow that some links are *not* contained in any cluster. Definition 4 defines this notion formally.

Definition *4:* A link e_{ij} is said to have *neutral membership* if and only if $\nexists E_c \in P''$ such that $e_{ij} \in E_c$.

Proposition *1:* Generalized link clustering supports the neutral membership of links.

To solve the generalized link clustering problem, we run a variation of SCAN on a link-space graph. The nodes classified as outliers or hubs in $LS(\mathcal{G})$ map to the neutral links of $\mathcal{G}$. Last, we would like to remark that any clustering algorithm can be adopted if it supports neutral membership.

C. Structural Clustering

Our variation for SCAN follows the basic definitions introduced in Section II-B except the rule of determining a core node. SCAN checks the *number* of similar neighbors whereas our variation the *fraction* of similar neighbors. The previous rule is problematic in our case since we are working on a transformed graph. Since the variation of the node degree gets larger in the link-space graph, using a threshold for the *number* of similar neighbors does not make sense. For example, in Figure 6, both nodes $v_{e_{ik}}$ and $v_{e_{jk}}$ had better be determined as core nodes when red links indicate similar neighbors. The previous rule, however, cannot handle this case unless the threshold is very low (e.g., 3), which is not suitable for high-degree nodes. In contrast, our new rule can handle this case by setting a proper *fraction* threshold (e.g., 0.5).

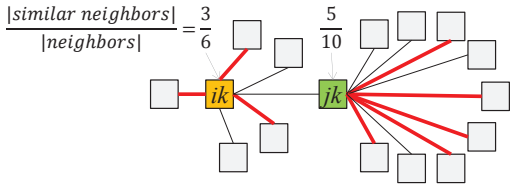


Fig. 6. An illustration of the rule of determining a core node.

In Definition 5, we formalize our new rule for selecting core nodes. This rule covers a broad range of scenarios including existing algorithms and our proposed algorithms.

Definition *5:* Let $f_v : [0,1]^{d_v} \to [0, d_v]$ be a function such that it maps a set of link similarities between $v \in V'$ and its neighbors into a value between 0 and d_v. Let $\{u_1, \ldots, u_{d_v}\}$ be a set of neighbors of v. For some $\epsilon \in [0,1]$, we call f_v a *core selection function* if $f_v(\omega(v,u_1), \ldots, \omega(v,u_{d_v})) = \sum_{j=1}^{d_v} \tau_{v,u_j} \cdot I(\omega(v,u_j) > \epsilon)$, where τ_{v,u_j} are nonnegative numbers and $I(\cdot)$ is an indicator function.

The core selection function f_v is a weighted sum of the indicator values for all incident links of v. If $\tau_{v,u_j} \equiv 1$, then it becomes identical to the core selection function used in DBSCAN and SCAN. If $\tau_{v,u_j} \equiv \frac{1}{d_v}$, then the core selection function considers the *fraction* of similar neighbors of v rather than the *number* of similar neighbors of v. Here, these coefficients normalize the importance for incident links of each node of the link-space graph. The rule with $\tau_{v,u_j} \equiv \frac{1}{d_v}$ is denoted by the *μ-fraction rule*. We use the μ-fraction rule in this paper since it leads to better results for the link-space transformation. Therefore, a node v is a *core* node if $f_v \geq \mu$ for some $\mu \in [0,1]$.

Note that our core selection function is computed locally, so it can be applied to large-scale networks or big data. In the μ-fraction rule, τ_{v,u_j}'s are the same for all neighbors u_j's. In fact, τ_{v,u_j} can be considered as the strength of the tie between the two nodes v and u_j. If such external information is available, τ_{v,u_j} can be set differently for each neighbor to better reflect the situation of the real world.

D. The algorithm LinkSCAN

LinkSCAN consecutively performs link-space transformation, structural clustering, and membership translation, which will be explained in the following subsections.

1) Link-Space Transformation: Algorithm 2 shows the procedure of transforming an original graph $\mathcal{G}$ into the link-space graph $LS(\mathcal{G})$. 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 if the corresponding links in $\mathcal{G}$ are incident to the common endpoint

Algorithm 2 LinkSpaceTransformation

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

1: /* create the nodes of $LS(\mathcal{G})$ */
2: **for** each $(v_x, v_y) \in E$ **do**
3: insert $v_{e_{(x,y)}}$ into V';
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 $(v_{e_{(z,w_1)}}, v_{e_{(z,w_2)}})$ 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 $(v_{e_{(z,w_1)}}, v_{e_{(z,w_2)}}) \in E'$ **do**
16: $\omega(v_{e_{(z,w_1)}}, v_{e_{(z,w_2)}}) = \frac{|\Gamma(v_{w_1}) \cap \Gamma(v_{w_2})|}{|\Gamma(v_{w_1}) \cup \Gamma(v_{w_2})|}$;
17: **end for**

(Lines 5∼13); and a weight is calculated using the links in $\mathcal{G}$ and is assigned to each link of $LS(\mathcal{G})$ (Lines 14∼17).

2) Structural Clustering: Algorithm 3 shows the procedure of clustering the nodes of the link-space graph, and the overall procedure is similar to SCAN. At each iteration, we randomly select a unclassified node v of a link-space graph (Line 4) and find a structure-connected cluster of v if it is a core node (Lines 5∼21). The node v will have neutral membership if it is *not* a core node (Lines 22∼24). The correctness of Algorithm 3 is asserted by the following theorem.

Theorem *1:* Under any core selection rule such that a node $v \in V'$ is a core node w.r.t. f_v with nonnegative τ_{v,u_j}'s, $\epsilon > 0$, and $\mu > 0$, there will be a unique community structure obtained by Algorithm 3.

Proof: (Sketch) When $\tau_{v,u_j} \equiv 1$ for $v \in V'$ and all $u_j \in N_\epsilon(v)$, the proof can be found in [8]. The structural reachability and connectivity used in SCAN can be directly extended to the case of any similarity measure between nodes. We show that it still holds for various nonnegative τ_{v,u_j}'s. Algorithm 3 decides whether a randomly selected node is a core or not at each iteration. Each decision depends only on the selected node $v \in V'$ and its corresponding core selection function f_v, because two global parameters ϵ and μ are fixed for every iteration. Hence, we can choose a set of core nodes deterministically regardless of the order of selecting nodes. The remaining part of the proof is the same as [8] since the output of the algorithm is determined by ϵ and μ when the core selection for all the nodes is done. ■

3) Membership Translation: This step restores the membership of a node from the result of link clustering. Simply, the membership of a link is mapped to that of its endpoints. Here, a node can belong to multiple communities if multiple links are incident to the node.

Algorithm 3 StructuralClustering

INPUT: a graph $LS(\mathcal{G}) = (V', E')$ and parameters ϵ and μ
OUTPUT: a disjoint clustering LC of $LS(\mathcal{G})$

1: Set clusterID to be 0; /* an initial id */
2: Mark all nodes in V' as *unclassified*;
3: $LS(\mathcal{G}) = (V', E')$ with the weight ω;
4: **for** each *unclassified* node $v \in V'$ **do**
5: **if** v is a core **then**
6: generate a new clusterID;
7: add $N_\epsilon(v)$ to a queue Q;
8: **while** $Q \neq 0$ **do**
9: $y \leftarrow$ the first node in Q;
10: $R = \{x \in V' | x$ is directly reachable from $y\}$;
11: **for** each $x \in R$ **do**
12: **if** x is *unclassified* or *neutral* **then**
13: /* x belongs to the current cluster */
14: assign the current clusterID to x;
15: **end if**
16: **if** x is *unclassified* **then**
17: insert x into Q;
18: **end if**
19: **end for**
20: remove y from Q;
21: **end while**
22: **else**
23: label v as *neutral*; /* neurtal membership */
24: **end if**
25: **end for**

4) Computational Complexity: The dominant part of LinkSCAN is structural clustering (Algorithm 3), and its running time is proportional to the number of links in $LS(\mathcal{G})$. We already discussed in Section II that each node of degree d_i of $\mathcal{G}$ forms a clique of size d_i (with $d_i(d_i - 1)/2$ links) in $LS(\mathcal{G})$. It implies that the number of links in $LS(\mathcal{G})$ is equivalent to $\sum_{i \in V} d_i(d_i - 1)/2$. Thus, the running time of LinkSCAN on a network $\mathcal{G} = (V, E)$ is $O(n\langle d^2 \rangle)$, where n is the total number of nodes and $\langle d^2 \rangle$ is the average value of the square of the degree of $\mathcal{G}$. Therefore, LinkSCAN runs in $O(n)$ for sparse networks.

E. The Algorithm LinkSCAN*

1) Sampling Strategy: LinkSCAN* additionally executes the sampling step before structural clustering. Since the running time of LinkSCAN is heavily dependent on the number of links, it is reasonable to reduce computational complexity by considering only a subset of links of the link-space graph $LS(\mathcal{G}) = (V', E')$. In Step 2 of Algorithm 1, for each node $v \in V'$, we take a random sample of size n_v from the set of incident links of v. After finishing sampling for every node, we construct a smaller network $LS'(\mathcal{G}) = (V', E'')$, where $E'' \subset E'$ such that E'' is the union of all sampled links on $LS(\mathcal{G})$.

A key issue is how to determine a proper sample size n_v, and we obtain it by Eq. (3), where α and β are nonnegative

constants used to control the sample size. Sampling is done only for the nodes having a sufficient number of incident links. That is, at a low-degree node whose degree is smaller than $\alpha + \beta \ln d_v$, its all incident links are sampled. Our heuristic method of determining these parameter values is discussed in Section III-F.

$$n_v = min\{d_v, \alpha + \beta \ln d_v\} \tag{3}$$

2) Error Bound: Sampling of links may introduce errors in the value of the core selection function f_v. In Theorem 2, we formally prove that the errors are *not* significant even when the sample size n_v is small. Theorem 2 also explains why we take the sample size by the form of Eq. (3).

Theorem *2:* For a node $v \in V'$, let f_v be a core selection function with $\tau_{v,u_j} \equiv 1/d_v$. Let $s = n_v$ be the sample size for the node v and $\{(v, u_1), \ldots, (v, u_s)\}$ be a set of the sampled links among the incident links of v. Let $X_v^{(i)} = I(\omega(v, u_i) > \epsilon)$ for each $i \in [1, s]$. Then, $\bar{X}_v = \frac{1}{s}\sum_{i=1}^{s} X_v^{(i)}$ is an estimate of f_v using the sampled incident links of v. The errors of this estimation are bounded by $Pr(|\bar{X}_v - f_v| \geq \delta f_v) \leq 1/d_v$ if $s \geq 3(\ln d_v + \ln 2)/(f_v \delta^2)$, for any constant $\delta > 0$.

Proof: Let us first introduce Lemma 1. This lemma explains that the sample mean of n independent trials is an estimator of p, where the probability that the absolute value of the bias is larger than δ times of p decreases exponentially as the sample size n increases.

Lemma *1:* (Chernoff bound [13]) Let $X_1, \ldots, X_n$ be independent Bernoulli random variables with the probability of success p. Then, for any $\delta \in [0, 1]$, $Pr(|\bar{X} - p| \geq \delta p) \leq 2e^{-np\delta^2/3}$, where $\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$.

We apply Lemma 1 to the proof of this theorem. Let's consider a procedure of choosing s links one by one. Replacing n, X_i's, and p by s, $X_v^{(i)}$'s, and f_v respectively, we obtain that $Pr(|\bar{X}_v - f_v| \geq \delta f_v) \leq 2e^{-sf_v\delta^2/3} \leq 1/d_v$ if $s \geq 3(\ln d_v + \ln 2)/(f_v\delta^2)$. ■

Theorem 2 says that the estimate $\bar{X}_v$ is very likely to be close to the true value f_v when we take a sample of size $3(\ln d_v + \ln 2)/(f_v\delta^2)$. The probability that $\bar{X}_v$ is estimated wrong is limited to be $1/d_v$, which is typically small. Meanwhile, the required sample size is much smaller than the node degree d_v especially at high-degree nodes, which is really desirable since it is more effective to reduce the computation cost for such high-degree nodes. For example, when $d_v = 1,000$, $f_v = 0.7$, and $\delta = 0.3$, then the bound is guaranteed if we use only about $1/3$ of all incident links of v. Since the Chernoff bound [13] is a bound for the worst case, in practice it is more reasonable to determine the sample size to be even smaller than $3(\ln d_v + \ln 2)/(f_v\delta^2)$, and in Section IV, we empirically confirmed that LinkSCAN* could still achieve very high accuracy. Overall, Theorem 2 provides the theoretical ground of the running-time enhancement of using sampling.

As a measure of the efficiency of the sampled links, we define the *sampling rate of links* by Definition 6. It increases monotonically from 0 to 1 as either α or β increases.

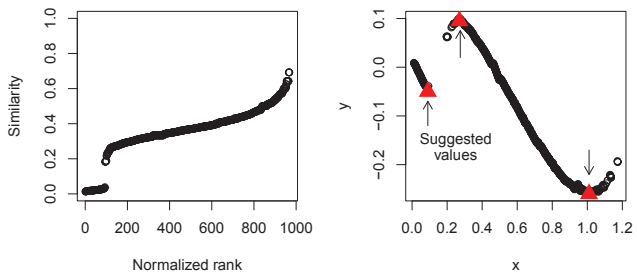
Definition *6:* The *sampling rate of links* is defined by the ratio of the number of the sampled links to the total number of links in $LS(\mathcal{G})$.

F. Parameter Selection

1) Parameters for Structural Clustering: LinkSCAN uses the two input parameters ϵ and μ. Our experience indicates that μ is less sensitive to the performance of LinkSCAN than ϵ. In addition, we have found by a set of extended simulations that our algorithm successfully identifies the clusters with $\mu = 0.7$. Thus, we decide to fix the value of μ to be 0.7 and to find the good estimates of ϵ.

Our heuristic method is similar to the one developed for DBSCAN [14]. First, we sample a certain fraction (or number) of the nodes of a link-space graph. For each sampled node v, we compute the 100μ-th percentile lowest value among link similarities between v and each of its neighbors. Then, we plot the curve for the 100μ-th percentile lowest similarities, sorted in descending order, as in Figure 7(a). The goal here is to find the values corresponding to the "knees" of the curve. Such knees are found in most cases, and the set of the nodes that belong to a cluster should change a lot around these values. Thus, the values are suggested for the estimates of ϵ.

The knees in the plot are detected in an automated way. We first normalize both axes and rotate the plot by 45 degrees clockwise. Figure 7(b) is obtained from Figure 7(a) in this way. Please note that the knees become more visible in Figure 7(b) than in Figure 7(a). Then, the next step is to find all *local extreme points* in this plot. For instance, there are three candidates for ϵ in Figure 7(b), and the set of these three values are provided to the algorithm.



(a) 100μ-th percentile similarities. (b) 45 degrees clockwise rotation.

Fig. 7. The 100μ-th percentile plot for parameter selection.

According to our extensive tests, we have found out that a good clustering is mostly achieved at the suggested values of ϵ, as we will show in Section IV-A. However, sometimes a good clustering is achieved at a small value of ϵ. Thus, when the number of suggested values is only one, in order to increase the chances of finding a good clustering, three or four values are randomly selected from the range $(0.0, 0.1)$ and are added to the set of suggested values. Since other algorithms usually recommend trying several parameter values and picking up the best result, trying these randomly-selected parameter values in our algorithm does not harm the fairness of comparison.

2) Parameters for Link Sampling: LinkSCAN* uses the two additional parameters α and β of Eq. (3) to control the sample size n_v. Determining a good sample size plays a crucial role in achieving both high efficiency and low errors. For too small a sample size, the errors will increase; for too large a sample size, the efficiency gain will decrease.

Our heuristic method is that the sample size should be at least the average degree of the link-space graph and increase slowly with respect to the node degree. The justification for "slow" increase is that a larger gap between a sample size and the node degree is allowed for higher-degree nodes according to Theorem 2. Thus, we will set α to be the average degree of the link-space graph and β to be a small constant.

For α, the *average degree of the link-space graph* is calculated by the first equality of Eq. (4), where $\langle d \rangle$ is the average degree of the original graph. If the degree distribution is non-skewed, e.g., the case of standard random graphs, the average degree of the link-space graph can be approximated by $2\langle d \rangle - 2$. Consequently, we recommend setting α to be $2\langle d \rangle$. For β, our experience indicates that LinkSCAN* shows the results closest to LinkSCAN with $\beta = 1$.

$$\frac{2\sum_{i \in V} d_i(d_i - 1)/2}{\sum_{i \in V} d_i/2} = 2(\langle d^2 \rangle / \langle d \rangle - 1) \approx 2\langle d \rangle - 2 \quad (4)$$

Surprisingly, $\alpha = 2\langle d \rangle$ with $\beta = 1$ performs well in all data sets in Section IV, whose degree distributions are diverse including power-law distributions. In our experiments, the NMI similarity measure between LinkSCAN and LinkSCAN* exceeds 0.9 even if the sampling rate of links is 0.5 ± 0.2 in synthetic networks and less than 0.3 in real-world networks.

IV. Experiments

In this section, we examine the performance of our proposed algorithms LinkSCAN and LinkSCAN*. We extensively tested our algorithms on both synthetic networks (Section IV-A) and real-world networks (Section IV-B). In all experiments, we set μ to be 0.7, α to be $2\langle d \rangle$, and β to be 1, as discussed in the previous section.

We compared the performance of five algorithms including our two algorithms. The three existing algorithms have been recognized as the state-of-the-art overlapping community detection algorithms, which will be explained in Section V.

(1) **LinkSCAN**: our proposed algorithm *without* sampling
(2) **LinkSCAN***: our proposed algorithm *with* sampling
(3) the CPM (Clique Percolation Method) [3], [1][1]
(4) the link-partition method [2], [4][2]
(5) the COPRA (Community Overlap PRopagation Algorithm) [6][3]

The weaknesses of these existing algorithms are that they do *not* perform well for (i) the networks with many highly overlapping nodes, (ii) the networks with various base-structures, and (iii) the networks with many weak ties. In contrast, we confirmed by extensive experiments on various networks that LinkSCAN and LinkSCAN* resolve those problems.

All experiments were conducted on Ubuntu Linux Servers with one CPU of Intel Xeon Processor E5620 and 96 GBytes of main memory. LinkSCAN and LinkSCAN* were implemented in C/C++ using the gcc compiler and the igraph library[4]; our heuristic method for parameter selection was implemented in R with the igraph library. For all other algorithms, we used the software packages provided by the authors, which are available on the Web.

A. Synthetic Networks

1) Data Generation: In order to demonstrate the performance of our proposed algorithms, we investigated the results on synthetic networks generated by the LFR benchmark [15]. It has been considered as a *de facto standard* model for generating networks with overlapping community structure that overcomes the drawbacks of the GN benchmark [16].

We analyzed the networks that have $1,000$ and $5,000$ nodes, which are the most common settings for the benchmark. The synthetic networks were built with varying the parameter values of degree of overlapping, average degree, community density, and so on. The parameters we used for the LFR benchmark are described in Table III.

TABLE III
PARAMETERS FOR THE LFR BENCHMARK (SELECTED).

Param.	Description
N	the number of nodes
$\langle k \rangle$	the average degree
μ_{mix}	the mixing parameter
O_n	the number of overlapping nodes
O_m	the number of communities per node

2) Evaluation Metrics: For the traditional graph partitioning problem, the *Normalized Mutual Information (NMI)* has been most widely used to measure the quality of partition when the ground-truth is known. Recently, Lancichinetti et al. [17] proposed an extended version of the NMI, which can be used for the case when a node may belong to more than one cluster. Note that the NMI is based on the information theory that compares the similarity between the memberships of two groups. Here, one group indicates the results of community detection, and the other group the ground-truth communities. Thus, the NMI naturally evaluates the quality of community detection. It ranges from 0 to 1 by normalization, and a higher value represents a better quality.

We also use the *F-score* that quantifies the performance of identifying *overlapping nodes*. For the ground-truth communities, we consider the vector of length N (number of nodes) where an element has a value 1 if the corresponding node belongs to multiple communities and 0 elsewhere. Another vector is constructed in the same way for the results of community detection. Then, comparing these two vectors gives an accuracy of identifying overlapping nodes.

[1] http://www.cfinder.org/
[2] http://barabasilab.neu.edu/projects/linkcommunities/
[3] http://www.cs.bris.ac.uk/~steve/networks/software/copra.html
[4] http://igraph.sourceforge.net/

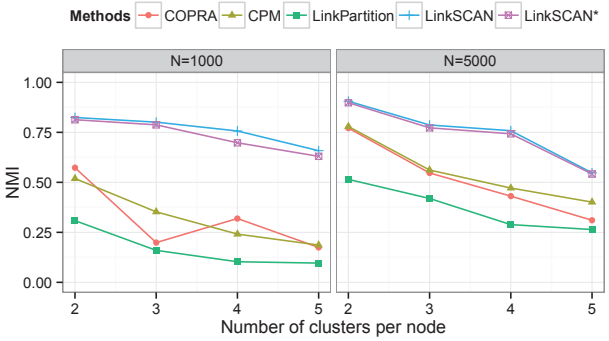


Fig. 9. Effects of the degree of overlapping ($\mu_{mix} = 0.1, O_n/N = 0.3, \langle k \rangle = 10$ and varying O_m from 2 to 5).

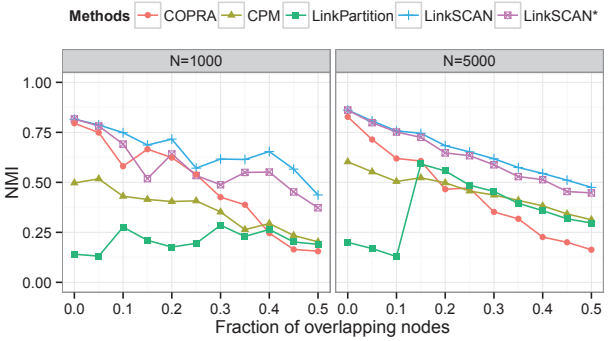


Fig. 10. Effects of the fraction of overlapping nodes ($\mu_{mix} = 0.1, O_m = 2, \langle k \rangle = 5$ and varying O_n/N from 0 to 0.5).

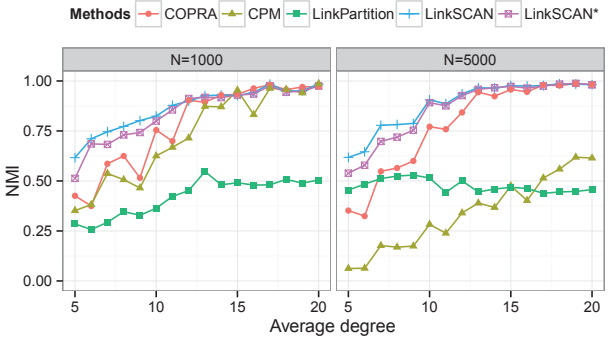


Fig. 11. Effects of the average degree ($\mu_{mix} = 0.1, O_m = 2, O_n/N = 0.3$ and varying $\langle k \rangle$ from 5 to 20).

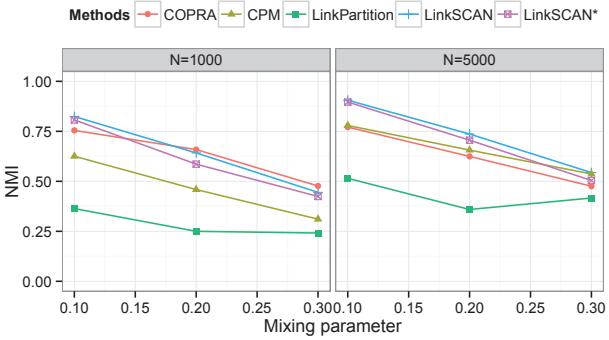


Fig. 12. Effects of the community density ($O_m = 2, O_n/N = 0.3, \langle k \rangle = 10$ and varying μ_{mix} from 0.1 to 0.3).

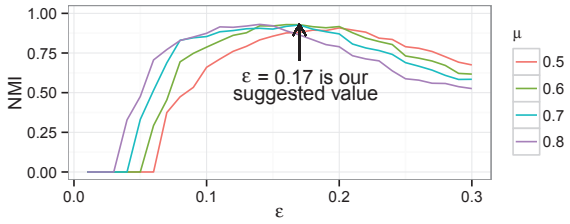


Fig. 8. NMI values when varying ϵ for a LFR benchmark graph.

3) Parameter Selection: We would like to show that our heuristic method for ϵ selection in Section III-F is robust and accurate. Figure 8 reports the NMI values when varying the value of ϵ with the four values of μ for a LFR benchmark graph. First, μ is less sensitive than ϵ since the highest NMI value can be achieved for every value of μ at a different value of ϵ. Thus, it is reasonable to fix the value of μ. Second, the NMI values at the ϵ's values suggested by our heuristic method are very close to the highest NMI value. In this figure, the NMI value at the suggested value $\epsilon = 0.17$ is 0.926, and the highest NMI value is 0.931. The same situation happened for other networks. Thus, we conclude that our heuristic method successfully finds the good parameter values.

4) Clustering Quality: We explain our results with the three scenarios below. The impact of each parameter is investigated with other parameters fixed at the typical values suggested by Xie et al. [18].

- *Networks with many highly overlapping nodes*: To generate *highly* overlapping nodes, we increased the value of O_m from 2 to 5. In Figure 9, LinkSCAN and LinkSCAN* perform well even if the average number of clusters per node is not small, e.g., 4 or 5. It is very natural that the accuracy goes down as the degree of overlapping increases. In addition, to generate *many* overlapping nodes, we increased the faction of overlapping nodes, which is O_n/N, from 0 to 0.5. Again, in Figure 10, our algorithms successfully identify the overlapping community structure when there exist many overlapping nodes, e.g., 40% or 50%. It is worthwhile to note that the performance gain increases as O_m or O_n/N increase especially when $N = 1,000$, i.e., when there are many overlaps relative to the network size.
- *Networks with various base-structures*: In general, it is hard to discover overlapping community structure for spare networks. However, as shown in Figure 11, LinkSCAN and LinkSCAN* perform well for a wide range of $\langle k \rangle$'s (the average degree). In particular for sparse networks, our algorithms do not fail to discover the reasonable structure even if there are many overlapping nodes, e.g., $O_n/N = 0.3$. The experiments show that the COPRA has a performance similar with our algorithms for the networks with the average degree larger than 10∼15. However, only our algorithms offer good performance for sparse networks.

In addition, LinkSCAN and LinkSCAN* give results better than or comparable to other algorithms for a wide range of μ_{mix} (the mixing parameter) as in Figure 12. It is the ratio of the number of inter-community links to the total number of links, and thus, it can be considered as the internal density of a community. Overall, we believe this result is meaningful in that our algorithms can discover communities of various internal density.

- *Networks with many weak ties*: Since there is no way of controlling weak ties by the LFR benchmark, we created two very simple networks in Figure 13. The color of a node represents the community membership generated by LinkSCAN and LinkSCAN*. A vacant node means that it does *not* belong to any community. It is obvious that our results are reasonable. However, other three algorithms failed even for these toy data sets.

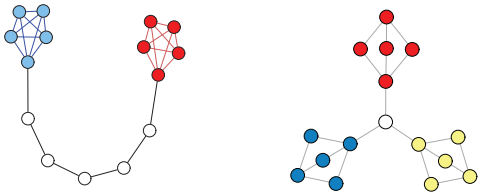


Fig. 13. Networks containing weak ties.

5) Overlapping Quality: Figure 14 shows the results of the F-score as the average degree increases. The parameters were configured as follows: $\epsilon = 0.07$ in both LinkSCAN and LinkSCAN*, $\nu = 9$ in the COPRA, and $k = 4$ in the CPM. Our algorithms consistently won at this setting for many highly overlapping nodes ($O_m = 5$ and $O_n/N = 0.4$). This figure indicates that the accuracy of identifying overlapping nodes in our algorithms is very high.

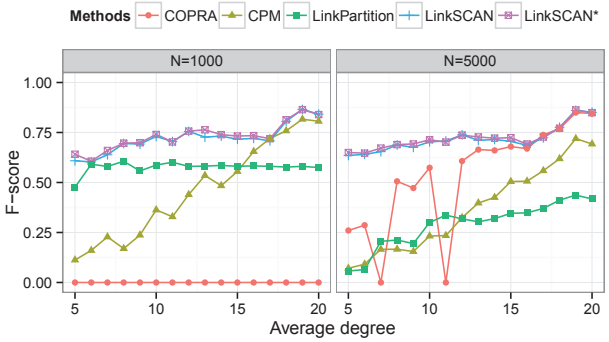


Fig. 14. Results of overlapping quality.

B. Real-World Networks

1) Data Sets: Table IV lists the real-world networks used for our experiments. Here, $\langle k \rangle$ and $\langle C \rangle$ are the average degree and clustering coefficient, respectively. For the network (a), we downloaded DBLP XML records and constructed a network such that each author becomes a node and two authors are connected if they coauthored a paper. For other networks, we downloaded them from the Web: (b), (c), and (d) from the Stanford Large Network Dataset Collection [5], (e) from the Facebook New Orleans Network [19], and (f) from Barabási's network databases [6].

TABLE IV
REAL-WORLD NETWORKS FOR EXPERIMENTS.

Name	# nodes	# links	$\langle k \rangle$	$\langle C \rangle$
(a) DBLP	1,068,037	3,800,963	7.50	0.19
(b) Amazon	334,863	925,872	5.53	0.21
(c) Enron-email	36,692	183,831	10.02	0.08
(d) Brightkite	58,228	214,078	7.35	0.11
(e) Facebook	63,392	816,886	25.77	0.15
(f) WWW	325,729	1,090,108	6.69	0.09

2) Evaluation Metrics: When the ground-truth community is unknown, the *modularity* is one of the most popular measures that evaluate the quality of clustering. Thus, we use the modularity measure M^{ov} of networks with overlapping communities, which was introduced by Lazar et al. [20]. For each cluster c_r, a $M^{ov}_{c_r}$ value is calculated such that it represents the normalized quantity of the difference between the inward-going links and the outward-going links. Then, the average value of $M^{ov}_{c_r}$'s is defined as the modularity with overlapping communities. It ranges from -1 and 1, and a higher positive value represents a better quality.

Intuitively, the M^{ov} value is larger if each community is more densely connected, and it could be higher if there are only a small number of nodes having a cluster membership. Thus, we need to consider two factors—quality and coverage—together to evaluate the performance of clustering when the ground-truth is unknown. For the former, the modularity is used. For the latter, we measure the fraction of nodes having a cluster membership [2], and the *clustering coverage* is denoted by CC.

3) Quality and Coverage: Figure 15[7] shows the results for the real-world networks. Each of M^{ov} and CC is normalized such that the minimum possible value is 0 and the best algorithm attains 1. When calculating CC, we removed trivial clusters whose members are less than 3. The sum of the two normalized values, which ranges from 0 to 2, is plotted in the figure. The best result was chosen for each algorithm after trying several parameter values suggested by the algorithms. The algorithm names are written by initials: COPRA by *CO*, CPM by *CP*, LinkPartition by *LP*, LinkSCAN by *LS*, and LinkSCAN* by *L**.

In Figure 15, either LinkSCAN or LinkSCAN* shows the best M^{ov} and $M^{ov} + CC$ values in each data set. Let's look into the main reasons for the poor modularity values of the existing algorithms in the DBLP network. The COPRA produced too many small-sized communities: those whose members were less than or equal to 3 took 74.6% of all the communities. The CPM and the link-partition method produced excessively overlapping communities: the number of

[5] http://snap.stanford.edu/data/

[6] http://www.nd.edu/~networks/resources.htm

[7] The CFinder software based on the CPM crashed for the Enron-email, Facebook, and WWW networks.

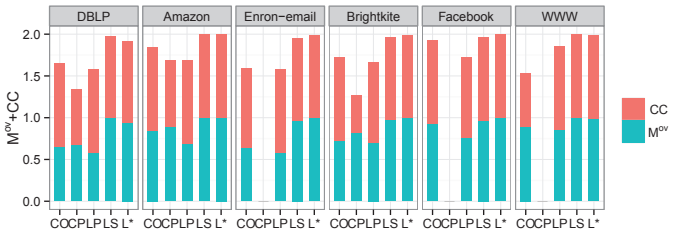


Fig. 15. Results for real-world networks.

the nodes that belonged to no less than 30 communities was 203 in the CPM and 66 in the link-partition method. On the other hand, the existing algorithms' high coverage is mainly due to one giant community containing almost all nodes.

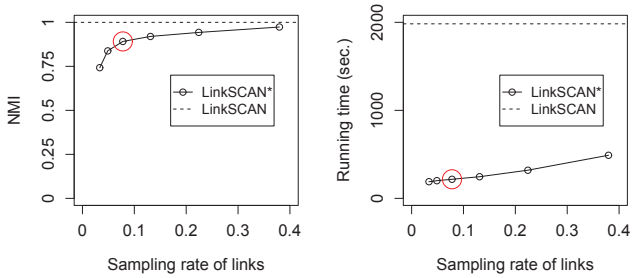
Overall, this result represents that LinkSCAN and LinkSCAN* discover meaningful community structures for a large fraction of real-world networks.

4) Case Studies: We have found several interesting cases which LinkSCAN and LinkSCAN* produce reasonable results but others do not. Here, we introduce only two cases because of the page limitation.

- In the CPM and the link-partition method, higher-degree nodes tend to overlap more excessively. It is known that *Paul Erdös* is one of the most prolific mathematicians. He had the habit of traveling from campus to campus: he did not stay long in one place and traveled back and forth among mathematical institutions until his death. This means that most links between Erdös and his collaborators are weak ties, and Erdös should be a *super-hub*. The CPM and the link-partition method detected 8 and 10 communities containing Erdös, respectively. In contrast, our algorithms and the COPRA concluded that he was *not* contained in any specific community, producing the correct result in our opinion.
- The CPM has a weak point in handling loosely-connected networks. In the DBLP network, if an author has written every paper with a small number of coauthors, he/she is not likely to be covered by the CPM even though he/she has several distinct coauthors. For example, *Prakash P. Shenoy* wrote 72 papers with 24 coauthors, and *Stephen Y. H. Su* 37 papers with 22 coauthors. They did not belong to any community in the CPM, whereas they properly belonged to one or more communities in our algorithms.

C. Comparison between LinkSCAN* and LinkSCAN

We compared LinkSCAN* with LinkSCAN when varying the sampling rate of links for the Enron-email data set. This case shows an ordinary performance among our real-world network data sets. Figure 16(a) reports the NMI values between the outputs of LinkSCAN* and LinkSCAN, and Figure 16(b) their running times. To control the sampling rate of links, α doubled from $0.5\langle d\rangle$ to $16\langle d\rangle$ while β was fixed to be 1. In general, the community detection result and running time of LinkSCAN* get closer to those of LinkSCAN as more links are sampled. When $\alpha = 2\langle d\rangle$ (marked by a red circle) according to our heuristics, using only less than 10% of links, the NMI similarity value exceeded 0.9, and the running time improved by about 10 times. Thus, LinkSCAN* is shown to significantly improve efficiency with introducing negligible errors compared with LinkSCAN.



(a) Accuracy (NMI). (b) Running time.

Fig. 16. Performance of LinkSCAN* for various sampling rates.

D. Scalability

We also checked the scalability of our proposed algorithms. We generated the LFR benchmark networks varying the number of nodes from $1,000$ to $1,000,000$ with the average degree fixed. The running time of LinkSCAN* for these networks is plotted in Figure 17, and it shows that LinkSCAN* has near-linear scalability. This result is encouraging and conforms to our complexity analysis in Section III-D. Hence, LinkSCAN* is shown to be efficient to support large-scale networks.

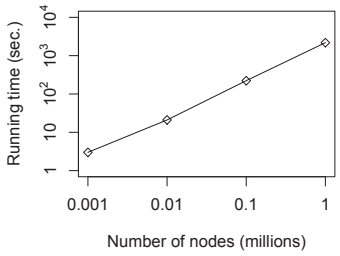


Fig. 17. Near-linear scalability of LinkSCAN*.

V. Related Work

In recent years, several clustering methods to uncover the overlapping community structure have been proposed [2], [4], [6], [21], [1], [22], [23]. The reader can get more details in a nice survey paper by Xie et al. [18]. We explain the three popular approaches compared in Section IV.

A. The Clique Percolation Method (CPM)

The CPM is a density-based technique, and it builds up the communities by taking each community as a maximal union of all k-cliques that are connected. Here, two k-cliques are connected if they share $k-1$ nodes. For $k = 2$, the CPM defines a community as a connected component of the network. For a suitable k, the members in a community obtained by the CPM can be reached through connected subsets of nodes, and the communities may share nodes with each other. However, it is sensitive to the parameter k; there is a trade-off between the quality of clustering and the coverage.

For instance, when k is large, the CPM finds communities well (high quality), but it covers only a small fraction of nodes (low coverage).

B. The Link-Partition Method

The link-partition method uses the similarity among relationships. It first converts a given graph $\mathcal{G}$ into the line graph $L(\mathcal{G})$, which is a graph such that 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 are adjacent in $\mathcal{G}$. Then, one can easily show that the node-partition of $L(\mathcal{G})$ maps one-to-one and onto (one-to-one correspondence) the link-partition of $\mathcal{G}$, which is an overlapping clustering for a set of nodes. Evans and Lambiotte [4] developed a link-to-link random walk, and Ahn et al. [2] proposed a Jaccard-type similarity to produce a partition of a set of links of the given network. These methods obtain the link communities and convert them to the node communities whose members can overlap. However, they tend to find unnecessary small clusters and may produce too many overlapping nodes.

C. The Label Propagation Method

The label propagation method also has been widely used for detecting communities in large complex networks. In this algorithm, all nodes propagate their labels to their neighbors for one step so that they reach a consensus on their community membership in many cases. Recently, an extension, named the Community Overlap PRopagation Algorithm (COPRA), to detect overlapping communities was proposed by Gregory [6]. They extended the label and propagate steps to include the information about more than one community. Owing to the simplicity of the procedure, the label propagation method is very fast, but it fails to converge in general [24].

VI. Conclusions

In this paper, we have presented a novel notion of the *link-space transformation*. This notion enables us to combine the advantages of both the original graph and the line graph, thereby conveniently achieving high-quality overlapping communities. Based on this notion, we have developed two overlapping community detection algorithms: *LinkSCAN* and *LinkSCAN**. The latter is an enhancement of the former for higher efficiency.

We have conducted extensive experiments using various synthetic and real-world networks. The results on the synthetic networks demonstrate that, in terms of the NMI values, our algorithms outperform existing algorithms especially for the networks with many highly overlapping nodes and those with various base-structures (e.g., a wide range of average degrees or community densities). Furthermore, for the real-world networks, our algorithms are shown to produce higher quality and coverage than existing algorithms.

Acknowledgment

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning (2012R1A1A1012954 and 2012R1A1A1014965).

References

[1] G. Palla, I. Derényi, I. Farkas, and T. Vicsek, "Uncovering the overlapping community structure of complex networks in nature and society," *Nature*, vol. 435, pp. 814–818, 2005.
[2] Y.-Y. Ahn, J. P. Bagrow, and S. Lehmann, "Link communities reveal multiscale complexity in networks," *Nature*, vol. 466, pp. 761–764, 2010.
[3] I. Derényi, G.Palla, and T. Vicsek, "Clique percolation in random networks," *Physical Review Letters*, vol. 94, p. 160202, 2005.
[4] T. S. Evans and R. Lambiotte, "Line graphs, link partitions, and overlapping communities," *Physical Review E*, vol. 80, p. 016105, 2009.
[5] S. Fortunato, "Community detection in graphs," *Physics Reports*, vol. 486, pp. 75–174, 2010.
[6] S. Gregory, "Finding overlapping communities in networks by label propagation," *New Journal of Physics*, vol. 12, p. 103018, 2010.
[7] M. Granovetter, "The strength of weak ties," *American Journal of Sociology*, vol. 6, pp. 1360–1380, 1973.
[8] X. Xu, N. Yuruk, Z. Feng, and T. A. J. Schweiger, "Scan: A structural clustering algorithm for networks," in *Proc. 13th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining*, 2007, pp. 824–833.
[9] 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)*, 2011, pp. 1661–1669.
[10] Y. Kim and H. Jeong, "Map equation for link communities," *Physical Review E*, vol. 84, p. 026110, 2011.
[11] J. Han, M. Kamber, and J. Pei, *Data Mining: Concepts and Techniques*, 3rd ed. Morgan Kaufmann, 2011.
[12] D. Liben-Nowell and J. Kleinberg, "The link-prediction problem for social networks," *Journal of the American Society for Information Science and Technology*, vol. 58, no. 7, pp. 1019–1031, 2007.
[13] H. Chernoff, "A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations," *Annals of Mathematical Statistics*, vol. 23, no. 4, pp. 493–507, 1952.
[14] 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*, 1996, pp. 291–316.
[15] A. Lancichinetti and S. Fortunato, "Community detection algorithms: a comparative analysis," *Physical Review E*, vol. 80, p. 056117, 2009.
[16] M. Girvan and M. E. J. Newman, "Community structure in social and biological networks," *Proc. National Academy of Sciences of the United States of America*, vol. 99, p. 7821, 2002.
[17] A. Lancichinetti, S. Fortunato, and J. Kertész, "Detecting the overlapping and hierarchical community structure in complex networks," *New Journal of Physics*, vol. 11, p. 033015, 2009.
[18] J. Xie, S. Kelley, and B. K. Szymanski, "Overlapping community detection in networks: the state of the art and comparative study," *ACM Computing Surveys*, vol. 45, no. 4, 2013.
[19] B. Viswanath, A. Mislove, M. Cha, and K. P. Gummadi, "On the evolution of user interaction in facebook," in *Proc. 2nd ACM Workshop on Online Social Networks (WOSN)*, 2006, pp. 613–622.
[20] A. Lázár, D. Ábel, and T. Vicsek, "Modularity measure of networks with overlapping communities," *Europhysics Letters*, vol. 90, no. 1, p. 18001, 2010.
[21] A. Padrol-Sureda, G. Perarnau-Llobet, J. Pfeifle, and V. Muntes-Mulero, "Overlapping community search for social networks," in *Proc. 26th Int'l Conf. on Data Engineering (ICDE)*, 2010, pp. 992–995.
[22] X. Wang, L. Tang, H. Gao, and H. Liu, "Discovering overlapping groups in social media," in *Proc. 8th IEEE Int'l Conf. on Data Mining (ICDM)*, 2010, pp. 569–578.
[23] J. Yang and J. Leskovec, "Community-affiliation graph model for overlapping network community detection," in *Proc. 10th IEEE Int'l Conf. on Data Mining (ICDM)*, 2012, pp. 1170–1175.
[24] L. Šubelj and M. Bajec, "Unfolding communities in large complex networks: Combining defensive and offensive label propagation for core extraction," *Physical Review E*, vol. 83, p. 036103, 2011.