BlackHole: Robust Community Detection Inspired by Graph Drawing

Sungsu Lim [†1], Junghoon Kim [‡2], Jae-Gil Lee [†3*]
[†] Department of Knowledge Service Engineering, KAIST
[‡] LG Electronics
Email: [1,3] {ssungssu, jaegil}@kaist.ac.kr, [2] junghoon00.kim@lge.com

***Abstract*—With regard to social network analysis, we concentrate on two widely-accepted building blocks: community detection and graph drawing. Although community detection and graph drawing have been studied separately, they have a great commonality, which means that it is possible to advance one field using the techniques of the other. In this paper, we propose a novel community detection algorithm for undirected graphs, called BlackHole, *by importing a geometric embedding technique from graph drawing*. Our proposed algorithm transforms the vertices of a graph to a set of points on a low-dimensional space whose coordinates are determined by a variant of graph drawing algorithms, following the overall procedure of spectral clustering. The set of points are then clustered using a conventional clustering algorithm to form communities. Our primary contribution is to prove that a common idea in graph drawing, which is characterized by consideration of *repulsive* forces in addition to attractive forces, improves the clusterability of an embedding. As a result, our algorithm has the advantages of being robust especially when the community structure is *not easily detectable*. Through extensive experiments, we have shown that BlackHole achieves the accuracy higher than or comparable to the state-of-the-art algorithms.**

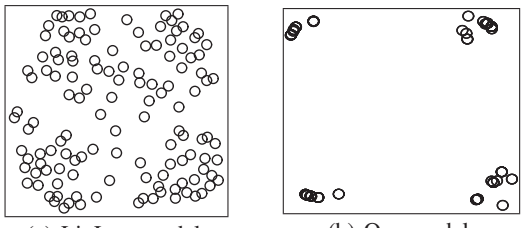
I. Introduction

Social network analysis is currently one of the most attractive issues in data mining and machine learning. More people are getting involved in social networks, and these social networks, in turn, become more complicated. The activities of the users on social networking services provide us with important clues to their real-world behaviors and relationships with others. Accordingly, social network analysis has emerged as a key technique in various disciplines including sociology, economics, epidemiology, politics, and psychology [1], [2], [3]. A lot of theories, models, and methods have been actively developed for this purpose. Among them, we would like to concentrate on two widely-accepted building blocks: community detection and graph drawing.

Community detection is a procedure of finding the community structure, with many edges joining vertices of the same community and comparatively few edges joining vertices of different communities [1]. Thus, communities are regarded as groups of vertices which likely share common properties and/or play similar roles within the graph. *Graph drawing* is a procedure of deriving a pictorial representation of the vertices and edges of a graph [4]. It often produces a node-link diagram in which vertices are represented as disks and edges as line segments or curves in the 2-dimensional space. Several quality measures have been defined for graph drawings, in an attempt to find an effective means of evaluating their aesthetics and usability.

Community detection and graph drawing have been parallel universes. Although it looks evident that these two problems have different objectives, they in fact have a great commonality. Graph drawing typically locates adjacent (*i.e.*, connected) vertices close with each other, to minimize edge crossings and eventually to optimize the aesthetic factor. As a result, a set of densely-connected vertices are gathered together, thereby naturally composing a community (*i.e.*, cluster) in the 2-dimensional space. Motivated by this commonality, we contend that several features inherent in graph drawing can advance community detection.

In this paper, we propose a novel community detection algorithm for undirected graphs, *by importing a geometric embedding technique from graph drawing*. Toward this goal, we develop a new model for graph drawing, which is tuned for community detection, not for visualization. Figure 1 shows the drawings of a graph by a force-directed layout model [5] and our new model respectively. In Figure 1(a), the vertices do not overlap with each other to consider visualization requirements. In Figure 1(b), by simply ignoring visualization requirements, the vertices of the same community are located at (almost) the same position. A position represents *multiple* vertices that possibly belong to the same community. Also, unlike a conventional model, we do not have to insist on a 2-dimensional space for layout. We note that, in Figure 1, our new model can reveal a community structure even for a complicated graph that a conventional model cannot.



(a) LinLog model. (b) Our model.

Fig. 1: Difference between a conventional drawing model and our model.[1]

Our community detection algorithm, which we call BlackHole, mainly consists of two phases in Figure 2. In the first phase, every vertex in a graph is mapped to a point in a low-dimensional space. Here, *multiple* points (*i.e.*, vertices) tend to collapse into a *single* position just like the black hole,

∗ Jae-Gil Lee is the corresponding author.

[1]The edges of the graph are omitted in the figure.

and this is why we named our algorithm BlackHole. In the second phase, the positions are grouped to form communities using conventional clustering algorithms such as k-means and DBSCAN [6]. Thus, a community is comprised of the vertices that are mapped to either the same position or a set of very close positions.

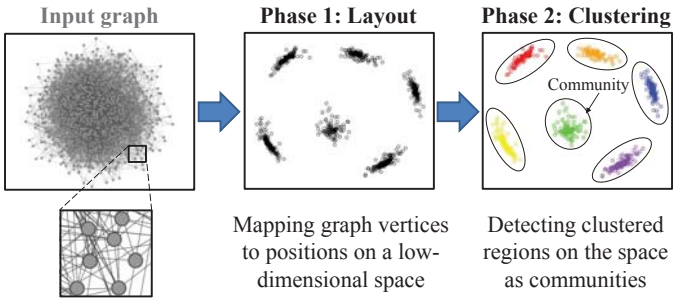


Fig. 2: The overall procedure of our algorithm BlackHole.

What is the advantage of BlackHole over existing community detection algorithms? BlackHole achieves the accuracy better than or comparable to the state-of-the-art algorithms such as Infomap [7], label propagation [8], Louvain [9], and spectral clustering [10]. This advantage becomes prominent when the community structure is *not easily detectable*. The *mixing parameter* is defined as the fraction of edges that are between different communities [11]. The higher the mixing parameter is, the less detectable the community structure is. The accuracy of the existing algorithms degrades drastically when the mixing parameter exceeds 0.5 [11], [12]. One of the main reasons for this poor accuracy is that most of the vertices are inclined to be wrongly assigned to very few gigantic communities [13]. BlackHole is shown to be much more robust to *high mixing* than the existing algorithms.

This advantage is important since high mixing occurs frequently as networks become more complicated, including in these two cases. First, the vertices lying at the boundary among communities play a role of mediation of the transition between communities [1]. In particular, when different communities share a commonality and cooperate with each other, the boundaries become more blurry. Such vertices near the boundaries tend to have more edges to other communities than to their own communities, thus causing high mixing. Second, there is a multi-aspect nature of interactions in network systems since vertices might be connected via multiple types of relationships at the same time [14]. Sometimes the aspects are invisible, and multiple types of relationships are aggregated and revealed in a single network. Then, the relationships irrelevant to community membership contribute to high mixing.

BlackHole is theoretically shown to be more robust to high mixing than two popular embedding-based community detection algorithms—spectral clustering and modularity optimization. BlackHole and spectral clustering are commonly translated to a divergence minimization problem [15] in which the main difference is the *type of divergence* determined by the way of handling *repulsive forces*. On the other hand, BlackHole and modularity optimization are commonly translated to a layout optimization problem in which the main difference is the *discreteness of dissimilarity*. These two differences enable us to achieve higher clusterability compared with the two algorithms. We empirically prove our claim also for other popular algorithms.

Overall, the contributions of this paper are summarized as follows.

1. We propose a novel paradigm for community detection inspired by graph drawing, thereby developing the algorithm BlackHole. The source code is available at https://github.com/jaegil/BlackHole. To the best of our knowledge, our work is the first attempt to show that graph drawing is *practically* usable for community detection.
2. We theoretically investigate the relationships between BlackHole and two embedding-based algorithms. The robustness of BlackHole to high mixing is interpreted using the relationships discovered.
3. We empirically show that BlackHole achieves higher accuracy than the state-of-the-art algorithms especially when a graph has high mixing of communities.

The rest of this paper is organized as follows. Section II reviews graph drawing and embedding. Section III proposes our community detection algorithm. Section IV formalizes the strength of our algorithm. Section V presents the evaluation results. Section VI summarizes related work. Finally, Section VII concludes this study.

II. Preliminaries

In this section, we explain the basics of graph drawing and embedding. Before proceeding, we summarize the notation used throughout this paper in Table I.

TABLE I: Summary of the notation.

Notation	Description
$\mathcal{G}$	an undirected weighted graph
d_v	the degree of a vertex v
w_v	a weight of a vertex v (*e.g.*, d_v)
$w_{u,v}$	a weight of an edge $\{u, v\}$
$p(v)$	a position of a vertex v by a layout p
$\mathcal{E}(p\|\mathcal{G})$	the energy with a layout p for $\mathcal{G}$
a, r	the parameters of the (a, r)-energy model
μ	the mixing parameter

A. Graph Drawing

For a given graph $\mathcal{G} = (V, E)$, *graph drawing* (or *graph layout*) is the problem of finding a set of positions (*i.e.*, locations) of the vertices and drawing a node-link diagram that visualizes the graph structure effectively. A *layout* $p : V \mapsto S$ is a function that maps from a set of vertices to a set of corresponding vertex positions in a drawing space S, *e.g.*, $S = \mathbb{R}^2$. Graphs drawn with existing algorithms tend to be aesthetically pleasing, exhibit symmetries, and produce crossing-free layouts based on their strategies. Among many approaches, we adopt *force-directed layout* algorithms owing to their simplicity and intuitiveness [4].

Those algorithms take both *attractive* and *repulsive* forces into account [16], [17], [5]. The former is seen as the attractive force between adjacent vertices connected via a spring, and the latter as the repulsive force between vertices (particles) electrically charged. Thus, adjacent vertices attract, which is inclined to group densely-connected vertices, and all pairs of vertices repulse, which is inclined to separate sparsely-connected vertices.

Formally, for a layout p and two vertices u, v ($u \neq v$), the *attractive force* exerted on u by v is denoted by Eq. (1), and the *repulsive force* exerted on u by v is denoted by Eq. (2) [18].

The strengths of the forces are often chosen to be proportional to some power of the distance.

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

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

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

The algorithms find an equilibrium state that the net force on each vertex becomes zero. This problem is equivalent to minimize the *energy* in Definition 1. The first term represents the sum of the attractive energies for every two adjacent vertices, and the second term represents the sum of the repulsive energies for every two vertices. Then, the energy is defined by subtracting the repulsive energies from the attractive energies. Graph drawing now translates to an optimization problem with the objective function Eq. (3) over all possible p's.

Definition 1: [18] The *energy* $\mathcal{E}(p|\mathcal{G})$ of a layout p for a graph $\mathcal{G}=(V,E)$ is given by Eq. (3).

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

Noack [18] has studied a general framework for the force-directed layout. The energy model in Eq. (3) is called the (a,r)-*energy model*. This model covers traditionally well-known graph drawing algorithms: the Fruchterman-Reingold algorithm [17] with $a=2, r=-1$, the Davidson-Harel algorithm [16] with $a=1, r=-3$, the LinLog algorithm [5] with $a=0, r=-1$, and so on. Note that the algorithms have different characteristics depending on the values of the two parameters a and r. In Section III-B, we choose the values of a and r so as to fulfill our purposes.

B. Embedding

An *embedding* is in general a map from a given space into another space. In order to explore the structure of a high-dimensional data set, embedding algorithms are used to determine a *low-dimensional representation* that preserves some interesting properties of the underlying data structure. The existing algorithms usually attempt to preserve global geometry or local geometry of a given data set.

Notably, *neighbor embedding* algorithms have been widely used for dimensionality reduction [15]. They find an embedding such that, for a given set of N data objects, the neighbors in the r-dimensional input space are approximately preserved in the s-dimensional output space (typically, $s \ll r$). More formally, $\sum_u D(\mathbf{x}_u\|\mathbf{y}_u)$ is minimized for a certain dissimilarity D, where $\mathbf{x}_u=(x_{u,v})_{v=1,\ldots,N}$ and $\mathbf{y}_u=(y_{u,v})_{v=1,\ldots,N}$ are probability distributions. Here, $x_{u,v}$ and $y_{u,v}$ are proportional to the probability that an object v is a neighbor of an object u *before* and *after* embedding respectively. Spectral clustering [19], [20], [10], which is a well-known algorithm for community detection in graphs, is basically neighbor embedding with the edge weight $w_{u,v}$ for a pair of vertices being considered as $x_{u,v}$.

The *information divergence* (or simply *divergence*) is a measure of dissimilarity between two probability distributions. Among several families of divergences, the β-*divergence* in Definition 2 is designed to work well for clustering [21], in which we are interested. It covers most commonly-used divergences: Itakura-Saito (IS) divergence with $\beta\to 0$, Kullback-Leibler (KL) divergence with $\beta\to 1$, and Euclidean distance with $\beta=2$.

Definition 2: [21] The β-*divergence* $D_\beta(\mathbf{x}_u\|\mathbf{y}_u)$ between two probability distributions $\mathbf{x}_u$ and $\mathbf{y}_u$ is given by Eq. (4).

$$D_\beta(\mathbf{x}_u\|\mathbf{y}_u)=\sum_v d_\beta(x_{u,v}|y_{u,v}),\ \text{where} \quad (4)$$

$$d_\beta(x_{u,v}|y_{u,v})=\frac{1}{\beta(\beta-1)}(x_{u,v}^\beta+(\beta-1)y_{u,v}^\beta-\beta x_{u,v}y_{u,v}^{\beta-1})$$

Yang *et al.* [15] proved that several neighbor embedding algorithms equivalently optimize the β-divergence. Variants of spectral clustering are equivalent to KL-divergence minimization, and the LinLog model is equivalent to IS-divergence minimization. In Section IV-A, we show that BlackHole can be unified into this framework of using the β-divergence, in order to formally investigate the connections between neighbor embedding algorithms and ours.

III. The Algorithm BlackHole

A. Community Detection

1) Problem Setting: The goal of *community detection* in graphs is to identify modules by only using the information encoded in the graph topology [1]. BlackHole performs this task, as described in Definition 3. It says that we discover *disjoint* communities.[2] In addition, the vertices are allowed *not* to be members of any community. That is, the union of the vertices in all communities may not be equivalent to V.

Definition 3: Given a graph $\mathcal{G}=(V,E)$, *community detection* finds the subgroups of vertices. The subgroups are called *communities*. Let C_1, C_2, ..., C_p be the communities found. Then, these communities satisfy the constraint, $C_i\cap C_j=\emptyset$ for $i\neq j$.

2) Overall Procedure: As shown in Figure 2, BlackHole consists of two phases: *layout* and *clustering*, which will be elaborated in Sections III-B and III-C respectively.

1. *Layout phase*: We map each vertex to a position in a low-dimensional space. For simplicity of exposition, we consider the 2-dimensional Euclidean space, but it can be generalized to any finite-dimensional space. The optimal layout is determined according to the (a,r)-energy model. A position to which *multiple* vertices are mapped is called a *black hole* as in Definition 4. A set of black holes found are provided to the clustering phase.
2. *Clustering phase*: We find the clusters of black holes using a conventional clustering algorithm. Here, we adopt DBSCAN [6], which is one of the most popular density-based clustering algorithms. A cluster of black holes composes a *community* as in Definition 5.

Definition 4: A *black hole* B_i is a position to which a subset of vertices $V_{B_i}\subseteq V$ are mapped.[3] The subset V_{B_i}

[2] Extension for overlapping community detection will be studied in our future work. Disjoint community detection can be used for overlapping community detection after transforming the original graph into a link-space graph, as proposed by Lim *et al.* [12].

[3] Practically, we regard a set of positions slightly different as the same by truncating coordinate values at some digit.

should have at least two vertices ($|V_{B_i}| \geq 2$). A vertex does not belong to multiple black holes ($V_{B_i} \cap V_{B_j} = \emptyset$ for $i \neq j$). The set of all black holes is denoted by $\mathcal{B}$.

Definition 5: A *community* C_j is the union of the vertices in a subset of black holes $\mathcal{B}_j \subseteq \mathcal{B}$. That is, $C_j = \bigcup V_{B_i}, \forall B_i \in \mathcal{B}_j$. A black hole does not belong to multiple communities ($\mathcal{B}_i \cap \mathcal{B}_j = \emptyset$ for $i \neq j$). The set of all communities is denoted by $\mathcal{C}$.

Algorithm 1 shows the pseudo code of BlackHole, which is self-explanatory.

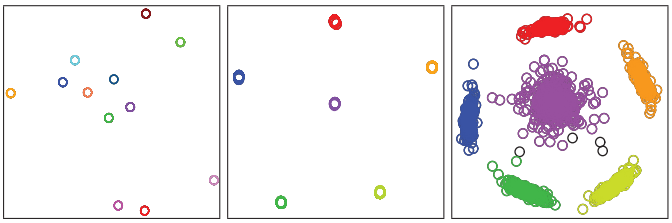
Algorithm 1 BlackHole

INPUT: An undirected weighted graph $\mathcal{G} = (V, E)$
OUTPUT: A set of communities $\mathcal{C} = \{C_1, C_2, \ldots, C_p\}$

```
/* PHASE I: LAYOUT */
/* Set the parameters for graph drawing */
a ← −0.95, r ← −1; /* Section III-B1 */
/* Execute Algorithm 2 in Section III-B2 */
B ← GraphDrawing(G, a, r);
/* PHASE II: CLUSTERING */
/* Set the parameters for density-based clustering */
ε, MinPts ← EstimateParam(B); /* Section III-C */
/* Execute the DBSCAN algorithm [6] */
C' ← DBSCAN(B, ε, MinPts);
for each C'_j ∈ C' do
    /* Compose a community from each cluster */
    newC ← {V_Bi | ∀B_i ∈ C'_j};
    C ← C ∪ {newC};
end for
return C /* a set of communities */
```

3) Example Results: Now, by presenting some example results, we would like to give the intuition on how BlackHole works.

Example 1: Figure 3 shows the results for three real-world[4] and synthetic networks. The two LFR networks [11] share the common properties except the mixing parameter μ. Here, a small circle represents a black hole, and a color identifies a community.



(a) Football. (b) LFR network 1. (c) LFR network 2.

Fig. 3: The results of BlackHole for three real-world and synthetic networks (in color).

(a) *American college football* (115 teams, 12 communities): Each of twelve black holes exactly forms a community.

(b) *LFR network 1* (50,000 vertices, 6 communities, $\mu = 0.3$): Each of six black holes exactly forms a community.

(c) *LFR network 2* (50,000 vertices, 6 communities, $\mu = 0.5$): Much more black holes are created in Figure 3(c) than in Figure 3(b), because the community structure is less detectable. Nevertheless, the black holes can be clearly grouped into six clusters, and each cluster exactly corresponds to a community. □

[4]This data set is available at http://www-personal.umich.edu/~mejn/netdata/.

B. Layout Phase

1) Model Selection: Since the (a, r)-energy model is general enough to represent many graph drawing models, we have determined to use it to find the layout suitable for community detection. The next task is to decide the best values of a and r. The rationale behind our decision is as follows.

a: We fix the value of a to -0.95. Ideally, the vertices of the same community should collapse into a single position. Motivated by *black holes in space*, the attractive force should become stronger as *connected* vertices get closer to each other. Figure 4 shows the trend of the attractive force depending on the value of a. When $a = -1$, the attractive force exponentially grows as the distance approaches zero. When larger values are used, it becomes weaker to prevent vertices from overlapping with each other. In addition, a cannot be less than or equal to -1 [18]. Thus, it is natural to choose a value slightly larger than -1, which is -0.95.[5]

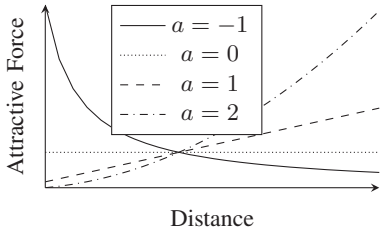


Fig. 4: Attractive force depending on a.

r: We fix the value of r to -1, since it is a typical value for many force-directed graph layout algorithms including the LinLog algorithm, the ForceAtlas algorithm [22], and the Fruchterman-Reingold algorithm. r is required to be smaller than a, and distances between communities are less dependent on densities for large $a - r$ [18]. A big negative value of r will make $a - r$ large. Thus, it is preferable to choose a value smaller than $a = -0.95$ and as large as possible, which is -1.

We examine the clustering tendency of the black holes while varying a with r fixed to -1. Since our graph drawing can be considered as micro clustering [23], the clustering phase will be less confused as clustering tendency is higher. The *Hopkins statistic* measures how far away a data set is from being uniformly distributed in the data space [24]. The closer it is to 0, 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 decrease as a approaches -1 and becomes almost 0 when $a = -0.95$. Overall, this examination confirms that our model selection with $a = -0.95$ and $r = -1$ is indeed correct for our purposes.

In summary, our objective function for energy minimization is Eq. (5), which is obtained by substituting a and r in Eq. (3) with -0.95 and -1 respectively. Here, $\|p(u) - p(v)\|^{-1+1}/(-1+1)$ is read as $\ln \|p(u) - p(v)\|$ because x^{-1} is the derivative of $\ln x$.

[5]We tested $a = -0.99$ as well, but the difference between $a = -0.95$ and $a = -0.99$ was negligible.

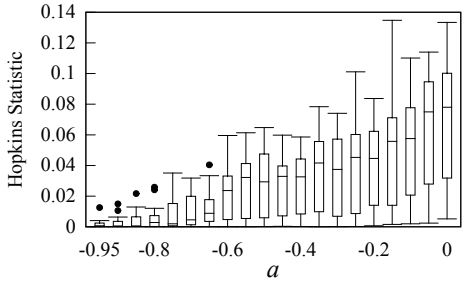


Fig. 5: Hopkins statistic depending on a.

$$\mathcal{E}(p|\mathcal{G}) = \sum_{\{u,v\}\in E} w_{u,v}\|p(u)-p(v)\|^{0.05}\cdot 20 - \sum_{\{u,v\}\in V^{(2)}} w_u w_v \ln\|p(u)-p(v)\| \quad (5)$$

2) Algorithm Descriptions: Algorithm 2 shows the pseudo code of our graph drawing (layout) algorithm, which places each vertex on the 2-dimensional space to derive a set of black holes. The algorithm first distributes the vertices arbitrarily on a plane (Lines 1∼4) and then keeps updating their positions toward a lower energy (Lines 5∼25). More specifically, for 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 the positions having a lower energy (Lines 21∼24). In our actual implementation, we divide Eq. (2) by the total weight of all vertices to prevent the vertices from spreading too much on a drawing space (Line 15).

Algorithm 2 Graph Drawing

INPUT: An undirected weighted graph $\mathcal{G}=(V,E)$
Parameter values of a and r
OUTPUT: A set of black holes $\mathcal{B}=\{p(v)|v\in V\}$
1: /* Generate initial positions of vertices */
2: **for each** $v\in V$ **do**
3: $p(v)\leftarrow Unif([-0.5,0.5]\times[-0.5,0.5])$;
4: **end for**
5: /* Minimize the energy by moving positions */
6: **repeat**
7: /* Construct a quadtree for a layout */
8: $T\leftarrow quadtree(\{p(v)|v\in V\})$;
9: /* Compute the net force acting on each vertex */
10: **for each** $v\in V$ **do**
11: **for each** u such that $\{u,v\}\in E$ **do**
12: $\overrightarrow{f}^{(a)}(v)\leftarrow \overrightarrow{f}^{(a)}(v)+$ Eq. (1); /* attractive */
13: **end for**
14: **for each leaf** $R_i\in T$ **do**
15: $\overrightarrow{f}^{(r)}(v)\leftarrow \overrightarrow{f}^{(r)}(v)+$ Eq. (2); /* repulsive */
16: **end for**
17: $\overrightarrow{f}(v)\leftarrow \overrightarrow{f}^{(a)}(v)+\overrightarrow{f}^{(r)}(v)$; /* net force */
18: **end for**
19: /* Choose a step size γ that minimizes Eq. (5) */
20: $\gamma\leftarrow argmin_{\gamma\in\{2^{-i}|i=0,\ldots,6\}}\mathcal{E}(p+\gamma\overrightarrow{f}|\mathcal{G})$;
21: /* Determine new positions */
22: **for each** $v\in V$ **do**
23: $p(v)\leftarrow p(v)+\gamma\overrightarrow{f}(v)$;
24: **end for**
25: **until** energy not decreasing
26: **return** $\mathcal{B}=\{p(v)|v\in V\}$ /* the set of black holes */

Calculating the repulsive energy requires calculating the distance for every pair of vertices. A straightforward approach would take $\mathcal{O}(|V|^2)$, which is quite expensive. To reduce this cost, we approximately calculate the repulsive force using a quadtree (Lines 8, 14∼16). A *quadtree* is a tree data structure in which each internal node has exactly four children [25]. The quadtree partitions a 2-dimensional space by recursively subdividing it into four regions or quadrants. Please refer to Figure 6 in which an example is shown. Then, the positions *in the same quadtree region* are approximated by a single position—*their center of mass*. That is, all vertices placed in the same region are regarded as a single vertex having the sum of their weights as its weight. As a result, the number of distance calculations reduces significantly, with the distances calculated still precisely.

Example 2: Figure 6 shows a quadtree with a set of five clustered positions. The entire region is divided into four regions $R1\sim R4$. The positions are concentrated around the center of mass, denoted by a solid circle, in $R1$, $R2$, and $R4$, whereas they are not in $R3$. Thus, $R3$ is divided into four regions once more. While traversing this quadtree, we include the repulsive force between a given vertex v and the current region in the total being accumulated. For example, the repulsive force exerted on v by all vertices in $R31$ is calculated collectively at once. □

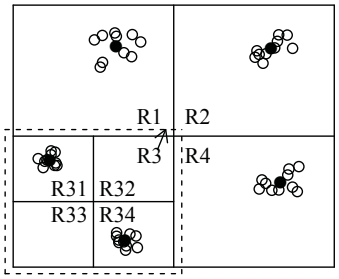


Fig. 6: A quadtree for a set of five clustered positions.

The extension for a 3-dimensional space is straightforward, simply by using the octree instead of the quadtree. However, for an even higher-dimensional space, the efficiency benefit is prohibited since the number of children in the index structure grows exponentially. Thus, it is *not* preferable to increase dimensionality unless the benefit in accuracy is significant. We discuss this issue in detail in Section V-C.

3) Approximation Accuracy: Approximation accuracy is dependent on how early we stop growing a quadtree (or octree). For a leaf of a quadtree (or octree), let s be the width of the region and d be the distance between the center of mass of the region and that of the entire region. Then, we stop growing a quadtree (or octree) if $s/d<\theta$ for a certain threshold θ. If s/d is small enough, the positions in a leaf can be approximated by their center of mass. Using this stop condition, the expected computational complexity reduces from $\mathcal{O}(|V|^2)$ to $\mathcal{O}(|V|\log|V|)$ as in Theorem 1.

Theorem 1: The computational complexity of Algorithm 2 is $\mathcal{O}(|E|+|V|\log|V|)$.

Proof: Computation of the attractive forces between adjacent vertices costs $\mathcal{O}(|E|)$. As for the repulsive forces, since we recursively traverse a quadtree (or octree) for each vertex, the computation takes $\mathcal{O}(|V|\cdot D)$ where D is the depth of

the tree. With our stop condition, Salmon [26] proved that, for the case of almost uniformly distributed particles, the expected value of D is $\mathcal{O}(\frac{1}{\theta^2}\log|V|)$ and it is $\mathcal{O}(\log|V|)$ when θ is a constant. Thus, the computational complexity of calculating the energy is $\mathcal{O}(|E|+|V|\log|V|)$. □

Corollary 1: The computational complexity of BlackHole (Algorithm 1) is still $\mathcal{O}(|E|+|V|\log|V|)$.

Proof: The computational complexity of DBSCAN is known to be $\mathcal{O}(|V|\log|V|)$ when a spatial index is used [6], and it is dominated by that of Algorithm 2. □

There is a tradeoff between approximation accuracy and speedup. A smaller value of θ makes a quadtree (or octree) deeper. Thus, as θ becomes smaller, approximation accuracy gets higher whereas speedup gets lower. The general trend with varying θ for many graphs is described in Figure 7. The left side indicates speedup, *i.e.*, the ratio of the elapsed time without approximation to that with approximation. The right side indicates approximation accuracy (NMI), *i.e.*, how close an approximate solution is to the real solution. As a result of balancing these two factors, the value of θ is determined to be 1 in our algorithm. This value also conforms to the suggested value in Barnes and Hut's work [27].

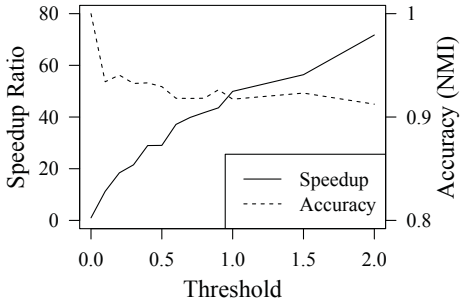


Fig. 7: Tradeoff between accuracy and speed.

C. Clustering Phase

Once the positions of black holes are obtained, it becomes possible to apply a conventional clustering algorithm to the positions on a low-dimensional space. Among many clustering algorithms, we have decided to adopt DBSCAN [6], which is one of the representative density-based clustering algorithms, since it fulfills our two requirements: the number of clusters is not known in advance, and the shape of a cluster is not necessarily circular.

The remaining task is to determine the values of two parameters: the size of a neighborhood (ε) and the minimum density of the neighborhood ($MinPts$). Overall, we follow the heuristic method suggested by the authors. A reasonable value of $MinPts$ is around $2\times$*dimensionality*, so $MinPts$ is set to be 5 for a 2-dimensional space and 7 for a 3-dimensional space. Then, to estimate the value of ε, we (i) compute the $(MinPts-1)$-th shortest distance for each of sampled vertices, (ii) plot a curve using the distances, sorted in descending order, (iii) normalize both axes and rotate the plot by 45 degrees counterclockwise, and (iv) find all local extreme points in this plot. These local extreme points are selected as the candidates for ε.

IV. Formalization and Discussion

In this section, we theoretically explore the relationships between BlackHole and existing representative algorithms. Since BlackHole is based on geometric embedding, we choose two popular embedding-based algorithms: *spectral clustering* [10] and *modularity optimization* [9]. Then, we translate our algorithm as well as each of them into a common framework and clarify the advantages of our algorithm over the two algorithms based on the differences identified.

A. Advantage over Spectral Clustering

1) Incorporation into Divergence Minimization: We first prove that, in Theorem 2, BlackHole can be incorporated into a framework of using the β-divergence.

Theorem 2: BlackHole is equivalent to an IS-divergence minimization problem.

Proof: We translate the objective function of BlackHole into the form of the IS-divergence in Eq. (6) defined by the limiting case of Eq. (4) for $\beta \to 0$.

$$\sum_{u\in V} D_{IS}(\mathbf{x}_u\|\mathbf{y}_u) = \sum_{\{u,v\}\in V^{(2)}} \left(\frac{x_{u,v}}{y_{u,v}} - \ln\left(\frac{x_{u,v}}{y_{u,v}}\right) - 1\right) \quad (6)$$

Eq. (5) is rewritten as Eq. (7) by the substitutions $x_{u,v} = w_{u,v}$ and $y_{u,v} = w_u w_v \|p(u)-p(v)\|^{-0.05}$. $\ln(w_u w_v/y_{u,v})$ is equal to $\ln(x_{u,v}/y_{u,v}) + \ln(w_u w_v/x_{u,v})$. Also, w_u, w_v, and $x_{u,v}$ are constant with respect to an embedding. Then, Eq. (8) is obtained from Eq. (7), where C is a constant.

$$\mathcal{E}(p|\mathcal{G}) = \sum_{\{u,v\}\in V^{(2)}} 20 w_u w_v \left(\frac{x_{u,v}}{y_{u,v}} - \ln\left(\frac{w_u w_v}{y_{u,v}}\right)\right) \quad (7)$$

$$= \sum_{\{u,v\}\in V^{(2)}} 20 w_u w_v \left(\frac{x_{u,v}}{y_{u,v}} - \ln\left(\frac{x_{u,v}}{y_{u,v}}\right) - 1\right) + C \quad (8)$$

Therefore, with Eqs. (6) and (8), the layout that minimizes the energies is equivalent to a graph embedding that minimizes a *weighted* sum of **IS**-divergences. □

A variant of spectral clustering, called *elastic embedding*, that minimizes Eq. (9) is shown to be equivalent to a **KL**-divergence minimization problem [15].

$$\sum_{\{u,v\}\in V^{(2)}} w_{u,v}\|p(u)-p(v)\|^2 + \lambda \sum_{\{u,v\}\in V^{(2)}} \exp(-\|p(u)-p(v)\|^2) \quad (9)$$

The second term causes a difference in the equivalent type of divergence. One may consider the second term as repulsive forces. However, its effect is limited since the value of the exponential function is bounded to 1, and it does *not* even exist in original spectral clustering. Thus, the reason why BlackHole and spectral clustering map to a different divergence is due to the *degree of considering repulsive forces*: relatively higher in BlackHole than in spectral clustering.

2) Effect of a Type of Divergence: Since both BlackHole and spectral clustering attempt to minimize a specific type of divergence—the IS-divergence in the former and the KL-divergence in the latter, we would like to examine the effectiveness of the divergence with respect to community detection. However, it is meaningless to *directly* compare the divergences of different types since they have different scales. Thus, our analysis adopts a *scaled* divergence score, which is the ratio of the divergence on a target graph to that on a baseline graph.

TABLE II: The notation solely used in Section IV-A.

Notation	Description
$\mathcal{G}_{Base}$, $\mathcal{G}_{Real}$	two graphs assumed in the analysis
D_{IS}, D_{KL}	the IS- or KL-divergence
$\mathcal{C}$	a set of communities $\{C_1, \dots C_p\}$
w_{C_i}	the total weight of the vertices in C_i
w_{C_i,C_j}	the total edge weight between C_i and C_j
X_ϵ	a random variable for variations in w_{C_i,C_j}
σ_ϵ	the standard deviation of X_ϵ

Such *baseline* and *target* graphs—denoted by $\mathcal{G}_{Base}$ and $\mathcal{G}_{Real}$ respectively—are generated using a simple generative model that adds edges within a community or across communities as designated by the parameters in Table II. For ease of theoretical analysis, we assume that $w_{C_i} = w_C$ for $1 \le i \le |\mathcal{C}|$ and $w_{C_i,C_j} = w_{C,C} + X_\epsilon$ where $X_\epsilon \sim N(0, \sigma_\epsilon^2)$ for $1 \le i < j \le |\mathcal{C}|$. Here, w_C and $w_{C,C}$ mean the common values regardless of communities. The only difference between the two graphs is whether $\sigma_\epsilon = 0\,(\mathcal{G}_{Base})$ or $\sigma_\epsilon > 0\,(\mathcal{G}_{Real})$. As shown by Figure 8, in unweighted graphs, $\mathcal{G}_{Base}$ has the same number of inter-community edges for every pair of communities whereas $\mathcal{G}_{Real}$ does not. Overall, $\mathcal{G}_{Base}$ is similar to an Erdős–Rényi graph [28], which is the most popular model for random graphs, since an edge between each pair of vertices across communities has the same probability, and $\mathcal{G}_{Real}$ is considered to represent a real-world graph having diverse weights of inter-community cuts.

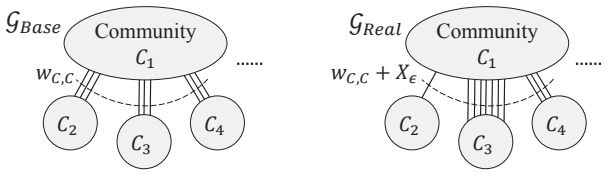


Fig. 8: Contrast between $\mathcal{G}_{Base}$ and $\mathcal{G}_{Real}$.

We now formally define the *divergence ratio* in Definition 6 for comparison between the two types of divergence. The denominator represents the standard score of that type of divergence, and the numerator is based on expectation since the definition of $\mathcal{G}_{Real}$ involves a random variable. The two input graphs are usually omitted in its notation. Just like the divergence, the divergence ratio also indicates the difference of the data structures represented on the original and embedded spaces. The smaller the divergence ratio is, the more effective that type of divergence is.

Definition 6: The *divergence ratio* is defined as Eq. (10). Here, D represents a specific type of divergence, and $D^*(\mathcal{G})$ is the *minimum* divergence value given a graph $\mathcal{G}$ and the divergence D.

$$DR(D; \mathcal{G}_{Real}, \mathcal{G}_{Base}) = \frac{\mathbb{E}[D^*(\mathcal{G}_{Real})]}{D^*(\mathcal{G}_{Base})}, \text{ where}$$
$$D^*(\mathcal{G}) = \underset{\{\mathbf{y}_u | u \in V\}}{\operatorname{argmin}} \sum D(\mathbf{x}_u \| \mathbf{y}_u) \tag{10}$$

In Theorem 3, using the notion of the divergence ratio, we prove that the IS-divergence used by BlackHole is *more effective for community detection* than the KL-divergence mainly used by spectral clustering.

Theorem 3: The divergence ratio with the IS-divergence is smaller than that with the KL-divergence, *i.e.*, $DR(D_{IS}) < DR(D_{KL})$.

Proof: We rewrite $D^*_{IS}(\mathcal{G}_{Base})$, $D^*_{IS}(\mathcal{G}_{Real})$, $D^*_{KL}(\mathcal{G}_{Base})$, and $D^*_{KL}(\mathcal{G}_{Real})$ as the functions of w_{C_i,C_j}. Then, they are approximated by the assumptions in $\mathcal{G}_{Base}$ and $\mathcal{G}_{Real}$. Through mathematical derivations, we have $\mathbb{E}[D^*_{IS}(\mathcal{G}_{Real})D^*_{KL}(\mathcal{G}_{Base}) - D^*_{IS}(\mathcal{G}_{Base})D^*_{KL}(\mathcal{G}_{Real})] < 0$. It implies that $DR(D_{IS}) < DR(D_{KL})$. See Appendix A for details. □

Theorem 3 theoretically implies the superiority of BlackHole to spectral clustering in the divergence perspective. Following this theorem, we also show that this superiority becomes more prominent in the two situations: i) as the *variation* of inter-community cuts increases (Corollary 2) and ii) as the *ratio* of inter-community cuts, which is proportional to the mixing parameter μ, increases (Corollary 3). It is worthwhile to note that Corollary 3 states that BlackHole is *robust to high mixing*.

Corollary 2: The value of $DR(D_{KL}) - DR(D_{IS})$ increases as σ_ϵ^2 (*i.e.*, $Var(X_\epsilon)$) increases.

Proof: See the proof of Theorem 3 in Appendix A. □

Corollary 3: The value of $DR(D_{KL}) - DR(D_{IS})$ increases as $w_{C,C}/w_C$ increases.

Proof: See the proof of Theorem 3 in Appendix A. □

B. Advantage over Modularity Optimization

In Theorem 4, we prove that BlackHole is closely related to the modularity optimization problem that maximizes *modularity* defined in Eq. (11). Here, C'_u and C'_v are the communities containing u and v respectively, and $\delta(\cdot)$ is the Kronecker delta that returns 1 if $C'_u = C'_v$ and 0 otherwise.

$$Q = \sum_{\{u,v\} \in V^{(2)}} \left(w_{u,v} - \frac{w_u w_v}{\sum_k w_k} \right) \delta(C'_u, C'_v) \tag{11}$$

Then, Theorem 4 says that modularity optimization can be considered as a 0-1 integer problem having a constraint that a distance between vertices must be either 0 or 1, but BlackHole is the same problem *except the constraint*.

Theorem 4: BlackHole is the linear programming relaxation of the 0-1 integer problem for modularity optimization.

Proof: Eq. (11) is rewritten as Eq. (12) by changing a maximization problem to a minimization problem and then substituting $1 - \delta(C'_u, C'_v)$ with $\|p(u) - p(v)\|$.

$$\sum_{\{u,v\} \in V^{(2)}} \left(w_{u,v} - \frac{w_u w_v}{\sum_k w_k} \right) (1 - \delta(C'_u, C'_v))$$
$$= \sum_{\{u,v\} \in V^{(2)}} \left(w_{u,v} \|p(u) - p(v)\| - \frac{w_u w_v}{\sum_k w_k} \|p(u) - p(v)\| \right) \tag{12}$$

$\|p(u) - p(v)\|$ is either 0 or 1 by the definition of the Kronecker delta, which can be satisfied by placing the vertices of a community at the same position from those whose pairwise distances are all 1's. Thus, by treating $\|p(u) - p(v)\|$ as a binary variable, modularity optimization is a 0-1 integer problem. If $a = r$, Eq. (12) has the same form as Eq. (3) except the constant factors $1/(a+1) = 1/(r+1)$ which change only the scaling of the optimal layouts. Since $a \to r$ in BlackHole, Eq. (12) also has approximately the same form as our objective

function Eq. (5) in which $\|p(u)-p(v)\|$ can be *any nonnegative value*. □

By Theorem 4, since the feasible region of modularity optimization is a subset of that of BlackHole, the optimal objective value of modularity optimization is always no better than that of BlackHole. Our algorithm in general will *not* satisfy all integer restrictions since it is preferable to locate the community positions depending on inter-community cuts. Therefore, this advantage tends to be more prominent when a graph has diverse weights of inter-community cuts, which is often accompanied by high mixing.

V. Experiments

We extensively tested the performance of BlackHole using not only real-world networks in Section V-A but also synthetic networks in Section V-B. We also tested the dimensionality effect and scalability of BlackHole in Sections V-C and V-D respectively. Seven community detection algorithms below were compared with one another. We included two variations of our algorithm depending on whether layout was done on a 2-dimensional space or a 3-dimensional space (1∼2). The baseline algorithm (3) is to combine a *conventional* graph drawing algorithm (Fruchterman-Reingold) and the k-means algorithm. The other four algorithms (4∼7) have been recognized as the state-of-the-art community detection algorithms [11].

1. BlackHole (**2D**)
2. BlackHole (**3D**)

} ← our proposed algorithm

3. Baseline [29] (based on conventional graph drawing)
4. Louvain [9] (based on modularity optimization)
5. Infomap [7]
6. Label propagation [8] (denoted as *LabelProp*)
7. Spectral clustering [19] (used parallel implementation)

In Section IV, we have already discussed spectral clustering and modularity optimization, in comparison to BlackHole. Here, we briefly explain the rest of the state-of-the-art algorithms used in the experiments. In *label propagation*, initially all vertices have different community labels, and each vertex updates its label from its neighbors by majority vote. *Infomap* finds a partition of vertices that minimizes the expected description length of a random walk. During the random walk, each step is always made to one of its neighbors. The common characteristic of the two algorithms is that they consider only the connections to *adjacent* vertices. Because each operation refers to only part of a graph, updating the labels or the partition may be stuck to a local optimum—an incorrect gigantic community—for the networks with high mixing.

Our experiments were *not* tuned for any specific algorithm or data set. For BlackHole, the two parameters of DBSCAN were simply configured by the heuristic explained in Section III-C. In other algorithms, every parameter was set to be the default value suggested by the authors. One exception was the number of communities required by k-means (in the baseline algorithm) and spectral clustering.[6] We used the exact number of communities for synthetic networks in order to favor the other algorithms. However, since we did not know it for real-world networks, we referred to the answers from either BlackHole or Louvain and chose a smaller number.

All experiments were conducted on Ubuntu Linux Servers with one CPU of Intel Xeon Processor E5-2670 and 96 GBytes of main memory. BlackHole and our heuristic method for parameter selection were implemented in C/C++ using the gcc compiler. For all other algorithms, we used the software packages provided by the authors and igraph library in R which are publicly available.

A. Real-World Networks

1) Data Sets: Table III lists the real-world networks used in our experiments. Here, $\langle k\rangle$ and $\langle C\rangle$ are the average degree and the average clustering coefficient respectively. We downloaded these networks from Stanford Large Network Dataset Collection[7] and Pajek Datasets[8]. Three of them are large enough, containing more than one million vertices. For the IMDb network, $\langle C\rangle = 0$ since it is a bipartite network with two types of vertices and hence has no triangles.

TABLE III: Real-world networks used for experiments.

	$\|V\|$	$\|E\|$	$\langle k\rangle$	$\langle C\rangle$
DBLP	317,080	1,049,866	6.62	0.632
Amazon	334,863	925,872	5.23	0.397
IMDb	1,324,748	3,792,390	5.73	0.000
Youtube	1,134,890	2,987,624	5.27	0.081
Skitter	1,696,415	11,095,298	13.08	0.258

2) Evaluation Metric: The ground-truth communities of real-world networks are usually unknown. Since there is no universal definition of a community, we decide to use a combination of various community-goodness measures introduced by Yang and Leskovec [30]: *internal density* (M1), *edges inside* (M2), *average degree* (M3), *fraction over median degree* (M4), *expansion* (M5), *cut ratio* (M6), *conductance* (M7), *normalized cut* (M8), and *flake out degree fraction* (M9), *i.e.*, those highlighted in Figure 9.[9] Here, an edge between measures indicates high correlation between them, and thus a group of the measures share certain characteristics. We note that the nine measures cover all three groups without a bias to a specific group. Since these measures are defined for a single community, for a set of communities, we use a weighted sum of the values where the weight of a community is the fraction of the vertices in the community.

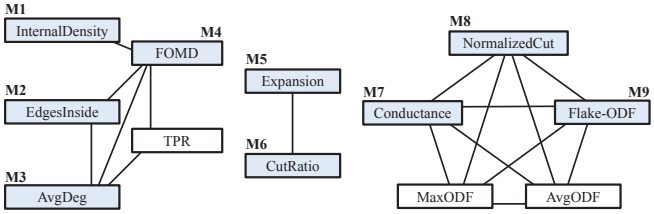


Fig. 9: Correlations of evaluation metrics [30].

We transform a value of a measure to a rank (1∼7) by that measure since M1∼M9 have different ranges. To prevent

[6] A popular heuristic is to find the largest gap between successive eigenvalues. However, we did not use the heuristic since it provided us with a unreasonably low number of communities (*e.g.*, 3 or 4) for the real-world networks.

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

[8] http://vlado.fmf.uni-lj.si/pub/networks/data/

[9] *Modularity* is excluded for a fair comparison since Louvain directly optimizes it.

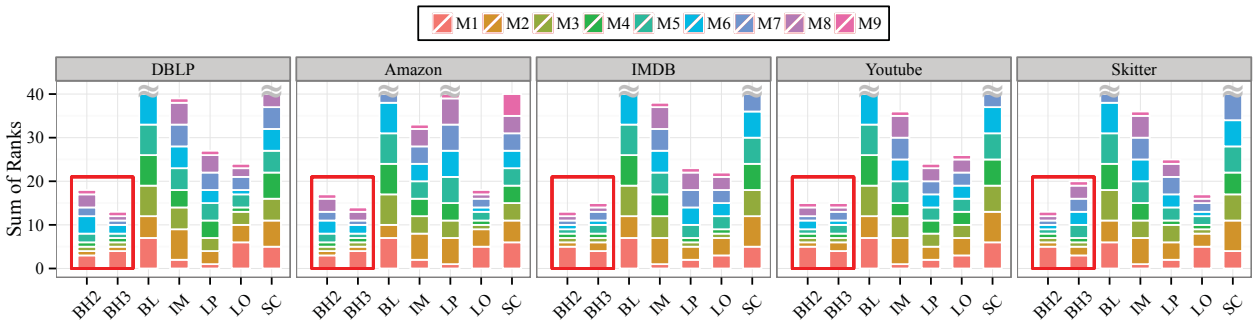


Fig. 10: Quality of community detection on real-world networks (in color).

the algorithms from being ranked differently by only a small difference, we allow them to have the same rank. Let's say that a value m_i of an algorithm is ranked at r_i. Then, the values $m_{i+j} (j \in \{1, 2, 3, 4, 5, 6\})$ for other algorithms are ranked at r_i if $|m_i - m_{i+j}|/m_{i+j} < 0.1$. These nine ranks are summed up to make an overall rank. Thus, the smaller the sum of ranks is, the better the performance is.

3) Results: Figure 10 shows the sum of ranks for each network data set. A rank by an individual measure is stacked up in each bar. The algorithm names are indicated by initials: BlackHole (2D) by "BH2," BlackHole (3D) by "BH3," Baseline by "BL," Louvain by "LO," Infomap by "IM," LabelProp by "LP," and Spectral Clustering by "SC." The sum of ranks over 40 is omitted in the plot.

It was observed that the two variants of BlackHole—*BH2* and *BH3*—had the best and second best cumulative (and average) ranks in four out of five data sets. In one exception (Skitter), only BH2 outperformed all other algorithms. Overall, our algorithm offered excellent performance for all networks, where the average ranks of BH2 are between 1.44 and 2.00 and those of BH3 are between 1.44 and 2.22. Louvain showed the performance next best to BlackHole. Spectral clustering, however, showed poor performance mainly because of a large number of communities (several hundreds or thousands) in those networks. We conjecture that the curse of dimensionality [31] kicked in, because the dimension of an embedding space for spectral clustering is as large as the number of communities. Baseline showed the worst performance since its layout had a weak tendency of clustering unlike ours.

These networks cover a wide range of clustering coefficients—from 0.081 (except IMDb) to 0.632. When the clustering coefficient of a network is low, the network is said to have a weaker community structure. Thus, we conclude that the performance of BlackHole is stable over the networks including those with a very low clustering coefficient.

B. Synthetic Networks

1) Data Sets: To more readily vary parameters and observe their effects, we also use the synthetic networks generated by the LFR benchmark [11], which allows us to generate the networks of various properties by controlling its parameters. The *vertex* information can be controlled by the number of vertices (N), the *edge* information by the average degree ($\langle k \rangle$) and the maximum degree ($maxK$), and the *community* information by the maximum community size ($maxC$) and the power-law exponent for the community size (t). Here, the size of a community is the number of the vertices belonging to the community; in addition, as t grows, the distribution of community sizes becomes more right-skewed. Last, the parameter of our main interest is the *mixing* parameter (μ).

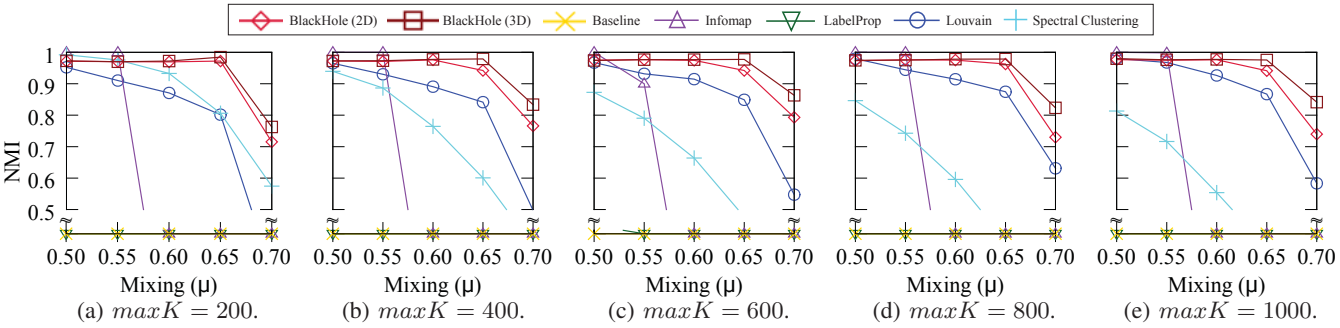
2) Evaluation Metric: In order to evaluate the accuracy of community detection, we use the *Normalized Mutual Information (NMI)* [32], which is one of the most widely used measures for community detection. It provides an information-theoretic measure for comparing different partitioning results and produces a value between 0 (disagreement) and 1 (agreement). In the experiments of this section, a higher NMI value indicates that the communities found by an algorithm match more with the ground-truth communities.

3) Results: The experiments using LFR benchmark networks were systematically conducted in three steps.

1. Varying the **mixing** and **edge** information (Figure 11)
2. Varying the **community** information (Figure 12)
3. Varying the **vertex** information (Figure 13)

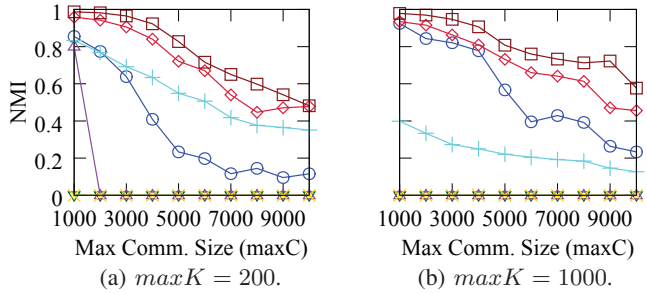
Figure 11 shows the effects of the mixing parameter (μ) when $N = 50,000$ and $\langle k \rangle = 40$. Each plot corresponds to the results for a different value of $maxK$, and we present the NMI values of the seven algorithms varying the value of μ. Since the NMI values of all algorithms except Baseline are over 0.95 when $\mu < 0.5$, we concentrate on the opposite side of μ, which is more difficult to handle. The results are summarized as follows.

- BlackHole in general achieved the best results. Also, by Corollary 3, BlackHole won against other algorithms including spectral clustering when mixing was high. The accuracy of BlackHole tended to be higher in a 3-dimensional space than in a 2-dimensional space, but the gap was not that significant (less than 10%).
- Louvain was the second best and competitive until μ was not so large. This close performance is due to the same form of the objective functions used by BlackHole and Louvain. However, when $\mu > 0.65$, the NMI value of Louvain dropped faster than that of BlackHole because of the limitation of discreteness described in Theorem 4.
- Spectral clustering identified meaningful communities. By Corollary 2, it got worse as $maxK$ increased, because a large value of $maxK$ would increase the variation of degrees and possibly the heterogeneity of mixing.
- Infomap and LabelProp failed to identify any community and hence made a single gigantic community when $\mu > 0.5$ or 0.6, as discussed earlier.
- Baseline failed to identify any community. Because conventional graph drawing typically spreads the vertices apart



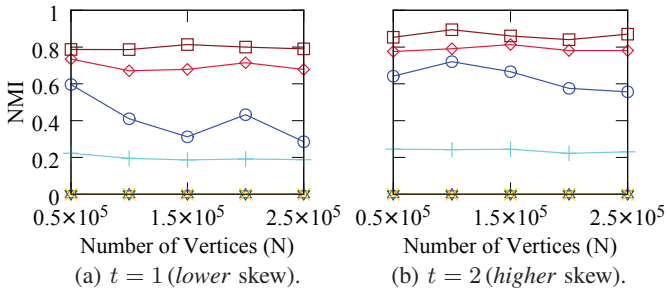
(a) $maxK = 200$. (b) $maxK = 400$. (c) $maxK = 600$. (d) $maxK = 800$. (e) $maxK = 1000$.

Fig. 11: Effects of the mixing parameter under various settings when $\langle k \rangle = 40$.



(a) $maxK = 200$. (b) $maxK = 1000$.

Fig. 12: Effects of the maximum community size.



(a) $t = 1$ (*lower* skew). (b) $t = 2$ (*higher* skew).

Fig. 13: Effects of the number of vertices.

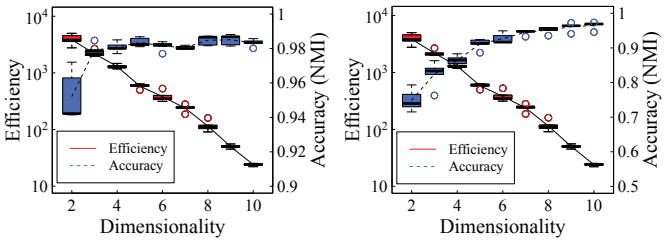
from each other, it is hard to identify the community structures in large-scale networks. Song and Bressan [29] tested this algorithm using very small networks with up to only 1,000 vertices.

Figure 12 shows the effects of the maximum community size ($maxC$) when $\mu = 0.7$ and $maxK = 200$ or $1,000$. Our intention here is to directly control the *variation of inter-community cuts*. Increasing $maxC$ would increase the variation of community sizes, thereby making the weights of inter-community cuts more diverse. As a result, the performances of all algorithms degraded as $maxC$ increased. Furthermore, a higher value of $maxK$ is likely to boost such variation. Thus, by Corollary 2, spectral clustering became worse in Figure 12(b) than in Figure 12(a). Overall, BlackHole was shown to be robust to high variations in considering that its NMI values were always highest and its decreasing rate was slower than or comparable to that of other algorithms.

Figure 13 shows the effects of the number of vertices (N) when $\mu = 0.7$ and $t = 1$ or 2. Our intention here is to expand a graph while preserving the other properties of the graph, in order to focus on the effects of N. For this purpose, we maintained the average number of communities by setting $maxC$ to be proportional to N. All algorithms were shown to be rather insensitive to N. In comparison between Figures 13(a) and 13(b), the overall trend was not affected when the skewness of community sizes intensified. As an exception, Louvain performed better in Figure 13(b) than in Figure 13(a), because it handled well small-sized communities prevalent with $t = 2$. Overall, the NMI values of BlackHole were kept to be quite high (around 0.8) for the entire range of N.

C. Dimensionality

To investigate the effect of the dimensionality on BlackHole, we repeated the experiments for Figure 11 while increasing the number of dimensions for layout until 10. We measured efficiency and accuracy for the *five* networks at the same number of dimensions. The results are reported for $\mu = 0.65$ and $\mu = 0.7$ in Figures 14(a) and 14(b) respectively. The *efficiency* is the number of vertices processed in a unit time, *i.e.*, (N/*elapsed time in minutes*), and the accuracy is measured by the NMI. As the number of dimensions increases, the accuracy tends to increase and converge at some point. In Figure 14(a), the accuracy reached the maximum early at the 3-dimension. The efficiency, however, degrades exponentially because the fan-out of the index structure doubles whenever the number of dimensions increases by 1. In summary, considering the trade-off between efficiency and accuracy, we have empirically found that a relatively small number of dimensions, *e.g.*, 3, is a reasonable choice unless $\mu \geq 0.7$.



(a) $\mu = 0.65$. (b) $\mu = 0.7$.

Fig. 14: Effects of the dimensionality.

D. Scalability

To check the scalability of BlackHole, we generated the LFR benchmark networks varying the number of vertices from $20,000$ to $1,280,000$ with the average degree fixed. Figure 15 illustrates the running time of BlackHole with the approximation technique and the other community detection algorithms. The performance of Baseline was omitted because it was very close to that of BlackHole (2D). The result shows that BlackHole has near-linear scalability, thereby confirming Corollary 1. Though BlackHole is not as fast as Louvain and LabelProp, it is sufficiently fast in considering that a large-scale network with over 1 million vertices can be processed

in 3∼4 hours on a single machine. In addition, our further investigation finds that the layout phase spends more than 90% of the total running time. We leave parallel processing of graph layout as future work.

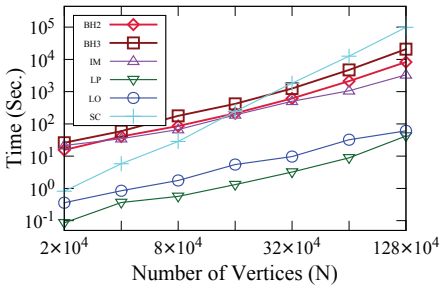


Fig. 15: Near-linear scalability of BlackHole.

VI. Related Work

A. Community Detection

Fortunato [1] and Schaeffer [2] conducted really extensive survey on community detection. There have been many techniques applied to detect communities: compression-based approaches such as Infomap [7] and diffusion-based approaches such as label propagation [8]. Another popular approach is optimizing a quality index such as modularity, and Louvain [9] belongs to this category. These existing algorithms often fail to detect communities and tend to produce a meaningless gigantic community for the networks with high mixing. Also, we are aware of the dynamical simplex evolution method [33] using the attractive and repulsive forces to determine the position of each vertex on a $(N-1)$-dimensional space, where N is the total number of vertices. However, as in Section V-C, the layout on such a high-dimensional space for a large-scale network is infeasible because of exponential computation cost. The authors tested a very small network with only 128 vertices. Song and Bressan [29] proposed an algorithm that combines a conventional graph drawing algorithm (Fruchterman-Reingold) and the k-means algorithm, but this baseline algorithm did not handle large-scale networks or high mixing, as shown in Section V-B.

B. Graph Drawing

Noack [5], [18] developed a general model for the force-directed layout, as discussed in Section II, and proved that it is closely related to community detection. Some special cases are equivalent to normalized cut minimization or modularity optimization. However, this theoretical study did not make a practical impact on community detection since the connection does not hold without the assumption that all vertices are exactly placed on either of two positions. In addition, conventional graph drawing models [16], [17], [5] do not provide very high clustering tendency and thus cannot be directly used for community detection. For example, in Figure 5, the LinLog model [5] with $a = 0$ showed lower clustering tendency than our model.

C. Spectral Clustering

Spectral clustering [19], [10] has a spirit similar to BlackHole in considering that vertices are mapped to points on a space and then these points are grouped by a conventional clustering algorithm. The main difference between them is how to map vertices onto the space. Spectral clustering finds the top-n eigenvectors of the Laplacian matrix of a given graph according to their eigenvalues. These eigenvectors form the coordinates of vertices on the n-dimensional space. The number of dimensions n is required to be as high as the number of clusters to find, unlike our algorithm. Finding eigenvectors is known to be computationally and memory intensive [1]. To resolve these drawbacks, several improvements based on sparsification [10], sampling [20], and parallelization [19] have been proposed recently. However, our experiments in Section V-A reported unsatisfactory accuracy especially for large-scale networks.

VII. Conclusion

In this paper, we proposed a novel community detection algorithm, which we call BlackHole. Motivated by the common nature between community detection and graph drawing, we explored the possibility of applying the techniques of graph drawing to community detection. Our new graph drawing model tries to place the vertices of the same community at a single position just like a black hole in space attracts everything nearby. Then, these positions are easily grouped by a conventional clustering algorithm. The lesson from graph drawing is that we should take not only attractive forces and but also *repulsive forces* into account. We theoretically proved the superiority of our algorithm to representative embedding-based algorithms and empirically verified our claims through extensive experiments. The experiment results are very promising. Our algorithm is shown to achieve very high performance regardless of the difficulty of community detection in terms of the mixing parameter, the clustering coefficient, and so on. If the community structure is *not easily detectable*, the strength of our algorithm becomes indeed prominent since other algorithms often fail to detect communities. Overall, we believe that our work is a step toward unifying the two different fields of social network analysis.

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 (2015R1A1A1A05001475).

References

[1] S. Fortunato, "Community detection in graphs," *Physics Reports*, vol. 486, no. 3, pp. 75–174, 2010.

[2] S. E. Schaeffer, "Graph clustering," *Computer Science Review*, vol. 1, no. 1, pp. 27–64, 2007.

[3] Z. Li, H. Xiong, and Y. Liu, "Mining blackhole and volcano patterns in directed graphs: a general approach," *Data Mining and Knowledge Discovery*, vol. 25, no. 3, pp. 577–602, 2012.

[4] R. Tamassia, *Handbook of Graph Drawing and Visualization*. Chapman and Hall/CRC, 2013.

[5] A. Noack, "Energy models for graph clustering," *Journal of Graph Algorithms and Applications*, vol. 11, no. 2, pp. 453–480, 2007.

[6] 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.

[7] M. Rosvall and C. T. Bergstrom, "Maps of random walks on complex networks reveal community structure," *Proc. Natl. Acad. Sci. USA*, vol. 105, no. 4, pp. 1118–1123, 2008.

[8] U. N. Raghavan, R. Albert, and S. Kumara, "Near linear time algorithm to detect community structures in large-scale networks," *Physical Review E*, vol. 76, no. 3, p. 036106, 2007.

[9] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2008, no. 10, p. P10008, 2008.

[10] U. Luxburg, "A tutorial on spectral clustering," *Statistics and Computing*, vol. 17, no. 4, pp. 395–416, 2007.

[11] A. Lancichinetti and S. Fortunato, "Community detection algorithms: A comparative analysis," *Physical Review E*, vol. 80, no. 5, p. 056117, 2009.

[12] 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*, 2014, pp. 292–303.

[13] I. X. Y. Leung, P. Hui, P. Lio, and J. Crowcroft, "Towards real-time community detection in large networks," *Physical Review E*, vol. 79, no. 6, p. 066107, 2009.

[14] S. Boccaletti *et al.*, "The structure and dynamics of multilayer networks," *Physics Reports*, vol. 544, no. 1, pp. 1–122, 2014.

[15] Z. Yang, J. Peltonen, and S. Kaski, "Optimization equivalence of divergences improves neighbor embedding," in *Proc. 31st Int'l Conf. on Machine Learning*, 2014, pp. 460–468.

[16] R. Davidson and D. Harel, "Drawing graphs nicely using simulated annealing," *ACM Trans. on Graphics*, vol. 15, no. 4, pp. 301–331, 1996.

[17] T. M. J. Fruchterman and E. M. Reingold, "Graph drawing by force-directed placement," *Software - Practice and Experience*, vol. 21, no. 11, pp. 1129–1164, 1991.

[18] A. Noack, "Modularity clustering is force-directed layout," *Physical Review E*, vol. 79, no. 2, p. 026102, 2009.

[19] W.-Y. Chen, Y. Song, H. Bai, and C.-J. Lin, "Parallel spectral clustering in distributed systems," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 33, no. 3, pp. 568–586, 2011.

[20] C. Fowlkes, S. Belongie, F. Chung, and J. Malik, "Spectral grouping using the Nystrom method," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 26, no. 2, pp. 214–225, 2004.

[21] A. Cichocki and S. Amari, "Families of alpha- beta- and gamma-divergences: Flexible and robust measures of similarities," *Entropy*, vol. 12, no. 6, pp. 1532–1568, 2010.

[22] 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*, vol. 9, no. 6, p. e98679, 2014.

[23] 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*, 1996, pp. 103–114.

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

[25] H. Samet and R. Webber, "Storing a collection of polygons using quadtrees," *ACM Trans. on Graphics*, vol. 4, no. 3, pp. 182–222, 1985.

[26] J. K. Salmon, "Parallel hierarchical n-body methods," Ph.D. dissertation, California Institute of Technology, 1991.

[27] J. Barnes and P. Hut, "A hierarchical $O(n \log n)$ force-directed calculation algorithm," *Nature*, vol. 324, no. 4, pp. 446–449, 1986.

[28] P. Erdős and A. Rényi, "On random graphs," *Publicationes Mathematicae*, vol. 6, pp. 290–297, 1959.

[29] Y. Song and S. Bressan, "Force-directed layout community detection," in *Proc. 24th Int'l Conf. on Database and Expert Systems Applications*, 2013, pp. 419–427.

[30] J. Yang and J. Leskovec, "Defining and evaluating network communities based on ground-truth," in *Proc. 10th IEEE Int'l Conf. on Data Mining*, 2012, pp. 1170–1175.

[31] S. Berchtold, B. Ertl, D. A. Keim, H.-P. Kriegel, and T. Seidl, "Fast nearest neighbor search in high-dimensional space," in *Proc. 14th IEEE Int'l Conf. on Data Engineering*, 1998, pp. 209–218.

[32] L. Danon, A. Diaz-Guilera, J. Duch, and A. Arenas, "Comparing community structure identification," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2005, no. 09, p. P09008, 2005.

[33] V. Gudkov, V. Montealegre, S. Nussinov, and Z. Nussinov, "Community detection in complex networks by dynamical simplex evolution," *Physical Review E*, vol. 78, no. 1, p. 016113, 2008.

APPENDIX
A. PROOF OF THEOREM 3

We assume that in an optimal layout all vertices in the same community are mapped to the same position. Let $d_{i,j}$ be the distance between the two positions for C_i and C_j.

(1) We calculate the optimal IS-divergence, using the objective function of BlackHole, and obtain Eq. (13).

$$\sum_{u,v}\left(20w_{u,v}\|p(u)-p(v)\|^{0.05}-\frac{w_u w_v}{\lambda}\ln\|p(u)-p(v)\|\right)$$
$$\cdots$$
$$=\sum_{i,j}\left(20w_{C_i,C_j}d_{i,j}^{0.05}-\frac{w_C^2}{\lambda}\ln d_{i,j}\right) \tag{13}$$

Differentiating Eq. (13) by $d_{i,j}$ and setting it to be 0, we have $d_{i,j}=(\lambda w_{C_i,C_j}/w_C^2)^{-20}$. By substituting the value of $d_{i,j}$ with it, the minimum IS-divergence is $\sum_u D_{IS}(\mathbf{x}_u\|\mathbf{y}_u)=\sum_{i,j}\frac{20w_C^2}{\lambda}\left(1-\ln\frac{w_C^2}{\lambda w_{C_i,C_j}}\right)=\sum_{i,j}(A+B\ln w_{C_i,C_j})$, where A and B are constant with respect to w_{C_i,C_j}.

(2) We calculate the optimal KL-divergence with the same substitutions as in Theorem 2 and obtain Eq. (14).

$$\sum_{u,v} x_{u,v}(\ln x_{u,v}-\ln y_{u,v})-x_{u,v}+y_{u,v}$$
$$\cdots$$
$$=\sum_u w_u(2\ln w_u+\ln\lambda-1)+\sum_{i,j}\left(-w_{C_i,C_j}\ln d_{i,j}^{-0.05}+\frac{w_C^2}{\lambda}d_{i,j}^{-0.05}\right) \tag{14}$$

In the same manner, we obtain the minimum KL-divergence $\sum_u D_{KL}(\mathbf{x}_u\|\mathbf{y}_u)=\sum_u w_u(2\ln w_u+\ln\lambda)-\sum_{i,j}w_{C_i,C_j}\ln(\lambda w_{C_i,C_j}/w_C^2)=C+\sum_{i,j}w_{C_i,C_j}\left(\ln(w_C^2/\lambda)-\ln w_{C_i,C_j}\right)$, where C is constant with respect to w_{C_i,C_j}.

In order to identify the effects of w_{C_i,C_j}'s, after excluding constant factors, we use $\sum_{i,j}\ln w_{C_i,C_j}$ and $\sum_{i,j}w_{C_i,C_j}\left(\ln(w_C^2/\lambda)-\ln(w_{C_i,C_j})\right)$ as the alternatives of the optimal IS-divergence and KL-divergence. We now obtain $D_{IS}^*(\mathcal{G}_{Base})$, $D_{IS}^*(\mathcal{G}_{Real})$, $D_{KL}^*(\mathcal{G}_{Base})$, and $D_{KL}^*(\mathcal{G}_{Real})$ by setting w_{C_i,C_j} according to the definitions of $\mathcal{G}_{Base}$ and $\mathcal{G}_{Real}$. In particular, $D_{IS}^*(\mathcal{G}_{Real})$ and $D_{KL}^*(\mathcal{G}_{Real})$ are approximated as Eqs. (15) and (16) using the Taylor series expansions up to order 2. Here, p denotes the total number of communities.

$$D_{IS}^*(\mathcal{G}_{Real}) \approx \binom{p}{2}\left(\ln w_{C,C}+\frac{1}{w_{C,C}}X_\epsilon-\frac{1}{2w_{C,C}^2}X_\epsilon^2\right) \tag{15}$$

$$D_{KL}^*(\mathcal{G}_{Real}) \approx \binom{p}{2}\left(w_{C,C}\left(\ln\frac{w_C^2}{\lambda}-\ln w_{C,C}\right)+\left(\ln\frac{w_C^2}{\lambda}-\ln w_{C,C}-1\right)X_\epsilon-\frac{1}{2w_{C,C}}X_\epsilon^2\right) \tag{16}$$

Finally, we obtain Eq. (17) since $\mathbb{E}[X_\epsilon]=0$ and $\mathbb{E}[X_\epsilon^2]=Var(X_\epsilon)+(\mathbb{E}[X_\epsilon])^2=\sigma_\epsilon^2$.

$$\mathbb{E}[D_{IS}^*(\mathcal{G}_{Real})D_{KL}^*(\mathcal{G}_{Base})-D_{IS}^*(\mathcal{G}_{Base})D_{KL}^*(\mathcal{G}_{Real})]$$
$$=-\sigma_\epsilon^2\left(\frac{\ln(\sqrt{\lambda}w_{C,C}/w_C)}{w_{C,C}}\right)\binom{p}{2}^2 \tag{17}$$

Eq. (17) <0 if $\lambda>(w_{C,C}/w_C)^{-2}$, where $w_{C,C}/w_C$ stands for mixing. Thus, it is satisfied if we take λ large enough. In practice, we take this value as the sum of weights of vertices for a given graph. Eq. (17) <0 is equivalent to $DR(D_{IS})<DR(D_{KL})$. In addition, $DR(D_{KL})-DR(D_{IS})$ should increase as Eq. (17) decreases, which does when σ_ϵ^2 or $w_{C,C}/w_C$ increase. □