Community detection methods for directed graphs

DANG Tien Dat, DO Duy Hieu and PHAN Thi Ha Duong*

Institute of Mathematics, Vietnam Academy of Science and Technology

Email: dat.dt173005@sis.hust.edu.vn (Dang Tien Dat), ddhieu@math.ac.vn (Do Duy Hieu), phanhaduong@math.ac.vn (Phan Thi Ha Duong).

Abstract

Community detection has been developed extensively with many different algorithms. One of the most powerful algorithms on undirected graphs is Walktrap, whose idea is to define the distance between vertices by using random walk and to evaluate the clusters by modularity function based on the degree of vertices. Although there are many directions to develop this method for directed graphs, none of those are effective. In this paper, we are interested in studying the Walktrap algorithm [28], the spectral method [25], and then extending them for directed graphs. We propose a new approach, in which the distance between vertices is defined by hitting time, and the modularity is computed based on the stationary distribution of a random walk. These definitions are very effective because of the development of algorithms about the hitting time and the stationary distribution so it is possible to compute them in good complexity. In particular, our proposed method can apply to directed graphs and the well-known results on undirected graphs are special cases. Besides, we also use the spectral method for these problems. And finally, we have also implemented our algorithms to demonstrate the plausibility and effectiveness of these methods.

1 Introduction

Community detection yields a wealth of insights into the latent relationships between vertices, which becomes even more important as networks are becoming extremely complex, where splitting vertices into different groups will be difficult. Some effective methods to solve community detection for undirected graphs are optimal modularity, hierarchical clustering and divisive clustering. The modularity function can be considered as a function to measure clustering quality, has up to two usages, can be considered as the objective function, or can be considered as the evaluation metric. Khan and Niazi [20] synthesized a variety of modularity functions for undirected cases such as weighted, unweighted; and non-ovelapping or overlapping for directed and undirected graphs. Newman and Girvan [26] stated a clustering criterion based on some centrality indices to find community boundaries, such as edge betweenness. Their paper takes a deeper look at using a random walk to define the distance between vertices, thereby opening up the distance between communities, by initially treating each vertex as a community and then consolidating it into a single community, finally finding the number of matching communities based on the selected metric. Clauset et al. [12] proposed a hierarchical clustering based on the modularity function which gives an algorithm in $O(n^2)$ time. In another direction, Newman [25] detailed how to use the modularity function as the objective function for undirected graphs

then use the spectral method to find the solution to divide the graph into two clusters. Some other methods in [9, 30, 3, 11] are also noteworthy.

Besides the successes of community detection methods for undirected graphs, several methods have been developed to directed graphs. Although Leicht and Newman [22] also proposed a modularity function for directed graphs, their definition of a new objective function lacks naturalness. Some other papers also study optimizing modularity on directed graphs [14], but their modularity functions have not mentioned detailed the internal structure of considered graphs.

1.1 Random walks on graphs

A random walk simulates a graph traversal in which the vertices traversed are chosen at random. To extend the result on undirected graphs to directed graphs, we will first define general directed graphs and then provide some notes on undirected graphs as a special case.

Let $G = (V, E, A)$ be a weighted digraph defined on the vertex set $V = \{1, 2, ..., n\}$, where A is a nonnegative but generally asymmetric weight matrix such that $a_{ij} > 0$ if and only if the directed edge (or arc) (i, j) belongs to E. We will refer to A simply as the adjacency matrix of G. For $i = 1, 2, ..., n$, we define the out-degree of vertex i, $d_i^+ = \sum_{j=1}^n a_{ij}$, and the in-degree of vertex i, $d_i^- = \sum_{j=1}^n a_{ji}$. In general, $d_i^+ \neq d_i^-$. However, we have $m = \sum_{i=1}^n d_i^+ = \sum_{i=1}^n d_i^-$.

For conciseness, in the following, unless otherwise stated, we refer to the out-degree of a vertex simply as its degree, and use d_i for d_i^+. Let D be a diagonal matrix of the vertex out-degrees, and define $P = D^{-1}A$. Then $P = [p_{ij}]_{i,j=\overline{1,n}}$ is the transition probability matrix of the Markov chain associated with a random walk on G, where at each vertex i, a random walk has probability $p_{ij} = a_{ij}/d_i$ of transiting from vertex i to vertex j if $(i, j) \in E$. We assume that G is strongly connected, namely, there is a directed path from each vertex i to every other vertex j.

We consider the aperiodic random walk X_k on a finite graph with n vertices. By the convergence theorem for finite Markov chains [2], the associated transition matrix P satisfies $\lim_{k\to\infty} P = P_\infty$, where $(P_\infty)_{ij} = \phi_j$, the j th component of the unique stationary distribution $\phi = (\phi_1, \phi_2, ..., \phi_n)$.

In the case of a undirected graph G where $a_{ij} = a_{ji}$, then $d_i = d_i^+ = d_i^-$, and $\phi_i = d_i/2m$ for all vertex i.

1.2 The Walktrap algorithm

The Walktrap method was introduced by Latapy and Pons [28] with the idea to define the distance between vertices by using the random walk process as follows.

Authors used the information given by all the probabilities P_{ij}^t to go from i to j in t steps.

Definition 1.1 *Let i and j be two vertices in the graph and*

$$r_{ij} = \sqrt{\sum_{k=1}^{n} \frac{(P_{ik}^t - P_{jk}^t)^2}{d_k}} = \|D^{-1/2}P_{i\bullet}^t - D^{-1/2}P_{j\bullet}^t\|. \tag{1.1}$$

where $\|\bullet\|$ is the Euclidean norm of $\mathbb{R}^n$.

This definition has two notable advantages: the first is that it accurately reflects the relationship between vertices when paying close attention to the transitions between vertices in the

graph; and the second is that it can be computed using the transition probability matrix's eigenvectors and eigenvalues. Based on the agglomerative hierarchical clustering method [16, 24], the Walktrap algorithm merges communities with the smallest distance. This method has been tested on real-world networks, particularly random partition graphs. The transition probability matrix must be symmetric in order to represent the Walktrap distance formula using eigenvalues and eigenvectors; however, this condition is only available on undirected graphs.

Many recent studies have tried to bring the matrix of a directed graph towards a symmetric matrix. Satuluri and Parthasarathy [31] defined a community as a group of vertices that share similar incoming and outgoing edges. Therefore, their approach can detect communities of vertices with homogeneous in-link and out-link structures (e.g., citation-based clusters), that do not necessarily share edges among them. More precisely, the authors proposed a two-stage framework that is compatible with the above discussion: (a) transformation to an undirected graph applying symmetrization methods to the adjacency matrix and (b) clustering the symmetrized graph using existing algorithms. The authors discussed and proposed various ways to symmetrize a directed network. The $A + A^T$ symmetrization method is one of the most simple method. Each directed edge will be replaced by one undirected edge. Then one can get an undirected connected graph G_U with the symmetric adjacency matrix $A_U = A + A^T$. Another symmetrization approach is based on the combination of the bibliographic coupling matrix $B = AA^T$ and the co-citation strength matrix $C = A^T A$, both these matrices are symmetric. Since both incoming and outgoing edges should be of the same importance for a clustering algorithm, the authors proposed to use the sum of these matrices as a symmetrization scheme:

$$A_U = AA^T + A^T A. \tag{1.2}$$

In other direction, symmetrization can be built based on random walks: the transformed graph is described by the following adjacency matrix:

$$A_U = \frac{\Phi P + P^T \Phi}{2}, \tag{1.3}$$

where $\Phi = diag(\phi_1, \phi_2, \ldots, \phi_n)$ is the diagonal matrix of stationary distribution.

In a similar spirit, Lai et al. [21] proposed a symmetrization method based on network embeddings. The basic idea is to embed the initial directed network into a vector space, preserving as much as possible from its local topological characteristics. The Laplacian, like the adjacency matrix, can be thought of as a network representation in Euclidean space that preserves similarities between vertex pairs. The idea is based on the extension of the Laplacian matrix for directed networks proposed by Chung [8].

$$L = \Phi - \frac{\Phi P + P^T \Phi}{2}. \tag{1.4}$$

Although many studies try to overcome the symmetric condition, these symmetrization methods often lose some of the meaning of directed graphs and is not a native extended version of the Walktrap algorithm for directed graphs.

1.3 Modularity Maximization

The basic definition of the community detection problem is to divide the vertices of a given network into groups that do not overlap, such that the density of connections inside the group

being higher than the density outside. The modularity function is built to evaluate the clustering quality. As mentioned above, the modularity function has two usages, it can be considered as an objective function and can also be considered as an evaluation metric, so there have been many studies on the modularity optimization method [13].

In this part, we present some main steps to optimize the modularity function. Various null models have been used, but the most common by far is the so-called configuration model [23, 24], a random graph model in which the degrees of vertices are invariant to match those of the observed network but edges are in other respects placed at random. The expected number of edges falling between two vertices i and j in the configuration model is equal to $d_i d_j/2m$. The actual number of edges observed to fall between the same two vertices is equal to the element A_{ij} of the adjacency matrix A, so that the actual-minus-expected edge count for the vertex pair is $A_{ij} - d_i d_j/2m$. Giving integer labels to the groups in the proposed network division and denoting by g_i the label of the group to which vertex i belongs, the modularity Q is then equal to

$$Q = \frac{1}{2m} \sum_{ij} \left[A_{ij} - \frac{d_i d_j}{2m} \right] \delta_{g_i g_j} \tag{1.5}$$

where δ_{ij} is the Kronecker delta. The leading constant $1/2m$ is purely conventional; it has no effect on the position of the modularity maximum.

M. E. J. Newman [25] used a Lagrange multiplier λ and proved that the maximum of Q is given by

$$As = D\lambda s. \tag{1.6}$$

It follows that

$$A^T s = \lambda D s \tag{1.7}$$

is equivalent to

$$P^T D s = \lambda D s. \tag{1.8}$$

Multiply on the left of both sides of the equation (1.8) by $D^{-1/2}$ and let $u = D^{1/2}s$, we have

$$D^{-1/2} P^T D^{1/2} u = \lambda u \tag{1.9}$$

Let $L = D^{-1/2} A D^{-1/2} = D^{1/2} P D^{-1/2}$, we have

$$L^T u = \lambda u. \tag{1.10}$$

Thus, to divide a graph into two clusters, we only need to find the eigenvector $v_\lambda = (v_1, v_2, ..., v_n)$ corresponding to the eigenvalue λ (usually is the second eigenvalue) of matrix L^T. Then, we split the cluster as follows: if v_i has a positive sign, then vertex i belongs to cluster 1, and if v_i has a negative sign, then vertex i belongs to cluster 2.

However, this algorithm applies only for undirected graphs.

1.4 Our contribution

In this paper, we are interested in studying the Walktrap algorithm [28], the spectral method [25] and then extending them for directed graphs. We propose a new approach, in which the distance between vertices is determined by hitting time H_{ij} - the expected number of steps taken by a random walk that starts at vertex i to reach vertex j.

Definition 1.2 *Let i and j be two vertices in the graph and*

$$r_{ij} = \sqrt{\sum_{k=1}^{n} \phi_k (H_{ik} - H_{jk})^2} = \|\Phi^{1/2} H_{i\bullet} - \Phi^{1/2} H_{j\bullet}\|. \tag{1.11}$$

where $\| \bullet \|$ is the Euclidean norm of $\mathbb{R}^n$.

On the other hand, the extension of the modularity function on an undirected graph toward directed graph is done in the following way. In the formula 1.5, d_i is not only the degree of the vertex i but it represents also the stationary distribution of a random walk on the undirected graph G. Then we can rewrite this formula as follows.

$$Q = \sum_{ij} \left[P_{ji} \frac{d_j}{d} - \frac{d_i}{d} \frac{d_j}{d} \right] \delta_{g_i g_j} = \sum_{ij} \left[P_{ji} \phi_j - \phi_i \phi_j \right] \delta_{g_i g_j}, \tag{1.12}$$

where $d = 2m$.

Now, consider a random walk on a directed graph G and apply the above formula to G. It is worth noting that, while formula 1.5 cannot be extended for digraph, formula 1.12 can. The main problem here is that for undirected graph, $\phi_i = \frac{d_i}{2m}$, which is easy for computing, but for directed graph, there is not a simple close formula for ϕ_i.

Our distance and modularity function definitions are very effective because the hitting time and stationary distribution calculation algorithms have been developed and have reached a good complexity. Furthermore, this method can be applied to directed graphs, with well-known results on undirected graphs acting as a special case. In addition, for these problems, we employ the spectral method. Finally, we implemented our algorithms to demonstrate the viability and efficacy of these methods.

2 Algorithm using hitting times

This part will present in detail our proposed distance between vertices. Besides, we also show the relationship between our distance formula and the original distance version. The relation with singular values in the directed case is also mentioned to suggest extension ideas in the future.

2.1 A distance r to measure vertex similarities

Because our proposed distance formula can be applied to both undirected and directed graphs, all notations of this subsection 2.1 are for the general graph.

Let a finite graph G, the hitting time H_{ij} from a vertex i to a vertex j, is the expected number of steps it takes for a random walk that starts at vertex i to reach vertex j. Note that when dealing with finite graphs, the hitting times from i to j is finite if and only if vertices i and are connected. We will use the information given by all the hitting times H_{ik}, from vertex i to reach every vertices k, this information resides in the n probabilities $(H_{ik})_{1 \leq k \leq n}$, which is nothing but the i^{th} row of the matrix H, denoted by $H_{i\bullet}$. To compare two vertices i and j using this data, we notice that two vertices of the same community tend to reach all the other vertices with the same expected number of steps. Thus if i and j are in the same community, we will probably have $H_{ik} \simeq H_{jk}$ for all $1 \leq k \leq n$. We can now give the definition of our distance between vertices, which takes into account all previous remark:

Definition 2.1 *Let i and j be two vertices in the graph and*

$$r_{ij} = \sqrt{\sum_{k=1}^{n} \phi_k (H_{ik} - H_{jk})^2} = \|\Phi^{1/2} H_{i\bullet} - \Phi^{1/2} H_{j\bullet}\|, \tag{2.1}$$

where $\| \bullet \|$ is the Euclidean norm of $\mathbb{R}^n$.

The random walk process can start at an arbitrary vertex of each community. From the formula 2.1 about distance between vertices, we propose the new distance between communities as the formula 2.3. We define the hitting times H_{Cj} the expected number of steps it takes for a random walk that starts from community C to reach vertex j.

$$H_{Cj} = \frac{1}{|C|} \sum_{i \in C} H_{ij}. \tag{2.2}$$

This defines a hitting times vector $H_{C\bullet}$ that allows us to generalize our distance:

Definition 2.2 *Let $C_1, C_2 \subset V$ be two communities. We define the distance $r_{C_1C_2}$ between these two communities by:*

$$r_{C_1C_2} = \sqrt{\sum_{k=1}^{n} \phi_k (H_{C_1k} - H_{C_2k})^2} = \|\Phi^{1/2} H_{C_1\bullet} - \Phi^{1/2} H_{C_2\bullet}\|, \tag{2.3}$$

where $\| \bullet \|$ is the Euclidean norm of $\mathbb{R}^n$.

Following [4] we introduce the fundamental matrix Z of P by

$$Z = (I - (P - P_\infty))^{-1}. \tag{2.4}$$

We will use the fundamental matrix to derive an expression for the expected hitting times from vertex i to vertex j. We have the following theorem.

Theorem 2.3 *([32]) The expected hitting times matrix H has entries given by*

$$H_{ij} = E_i T_j = \frac{z_{jj} - z_{ij}}{\phi_j}. \tag{2.5}$$

From Theorem 2.3, we have

$$H_{ik} = \frac{z_{kk} - z_{ik}}{\phi_k} \text{ và } H_{jk} = \frac{z_{kk} - z_{jk}}{\phi_k}, \tag{2.6}$$

it follows that

$$H_{ik} - H_{jk} = \frac{z_{kk} - z_{ik}}{\phi_k} - \frac{z_{kk} - z_{jk}}{\phi_k} = \frac{z_{jk} - z_{ik}}{\phi_k}. \tag{2.7}$$

Putting this formula into equation (2.1) of definition of r, we have

$$r_{ij} = \sqrt{\sum_{k=1}^{n} \frac{(z_{ik} - z_{jk})^2}{\phi_k}} = \|\Phi^{-1/2} Z_{i\bullet} - \Phi^{-1/2} Z_{j\bullet}\|. \tag{2.8}$$

2.2 Relation with spectral approaches on Undirected Graphs

Given a undirected graph G with a aperiodic transition matrix P. Conjugating P by $D^{1/2}$ transforms P into a symmetric matrix, this is because

$$S = D^{1/2}PD^{-1/2} = D^{-1/2}AD^{-1/2}. \tag{2.9}$$

As this new matrix S is symmetric, it is diagonalizable in the form

$$S = W\Sigma W^T, \tag{2.10}$$

where Σ is the diagonal matrix consisting of the eigenvalues of S, $\lambda_1, \ldots, \lambda_n$, and W is the associated matrix of orthogonal eigenvectors (columns $w_1, \ldots, w_n$). Substitution shows that $\lambda_1 = 1$ is an eigenvalue of S with associated eigenvector $w_1 = (\sqrt{\phi_1}, \ldots, \sqrt{\phi_n})$. Now let

$$v_\alpha = D^{-1/2}s_\alpha \text{ and } u_\alpha = D^{1/2}s_\alpha. \tag{2.11}$$

As P and S are similar, they have the same eigenvalues, and the calculation

$$Pv_\alpha = PD^{-1/2}w_\alpha = \lambda_\alpha D^{-1/2}w_\alpha = \lambda_\alpha v_\alpha, \tag{2.12}$$

shows that v_α is a right eigenvector of P corresponding to λ_α. A similar calculation shows that the u_α's are the left eigenvectors of P. Therefore, $P = V\Sigma U^T$, or in particular, (2.11) implies that $u = Dv$,

$$P = V\Sigma V^T D. \tag{2.13}$$

We need the following theorem.

Theorem 2.4 *([32]) Let P be the transition matrix for an aperiodic random walk, expressed as in (2.13). Then the fundamental matrix Z can be diagonalized in the following form:*

$$Z = V\bar{\Sigma}V^T D, \tag{2.14}$$

where, $\bar{\Sigma} = (\bar{\lambda_1}, ..., \bar{\lambda_n})$, $\bar{\lambda_1} = 1$, and for $i > 1$, $\bar{\lambda_i} = (1 - \lambda_i)^{-1}$.

Theorem 2.5 *The distance r is related to the spectral properties of the matrix P by:*

$$r_{ij}^2 = d\sum_{\alpha=2}^{n} \frac{1}{(1-\lambda_\alpha)^2}(v_\alpha(i) - v_\alpha(j))^2, \tag{2.15}$$

where $(\lambda_\alpha)_{1\leq\alpha\leq n}$ and $(v_\alpha)_{1\leq\alpha\leq n}$ are respectively the eigenvalues and right eigenvectors of the matrix P. Furthermore, we can rewrite the distance formula as follows:

$$r_{ij}^2 = d\sum_{\alpha=2}^{n} \frac{1}{\sigma_\alpha^2}(v_\alpha(i) - v_\alpha(j))^2, \tag{2.16}$$

with $v_\alpha = D^{-1/2}s_\alpha$, where $(\sigma_\alpha)_{1\leq\alpha\leq n}$ and $(s_\alpha)_{1\leq\alpha\leq n}$ are respectively the eigenvalues and right eigenvectors of the normalized Laplacian matrix $\mathcal{L}$.

Proof From Theorem 2.4, we have

$$Z = V\bar{\Sigma}V^T D = \sum_{\alpha=1}^{n} \bar{\lambda_\alpha} v_\alpha u_\alpha^T. \tag{2.17}$$

It follows that

$$Z_{i\bullet} = \sum_{\alpha=1}^{n} \bar{\lambda_\alpha} v_\alpha(i) u_\alpha = D^{1/2} \sum_{\alpha=1}^{n} \bar{\lambda_\alpha} v_\alpha(i) s_\alpha.$$

We put this formula into equation (2.8). Then we use the Pythagorean theorem with the orthonormal family of vectors $(w_\alpha)_{1\leq\alpha\leq n}$, while keeping in mind that the vector v_1 is constant to remove the case $\alpha = 1$ in the sum. Finally we have

$$r_{ij}^2 = d\sum_{\alpha=2}^{n} \bar{\lambda_\alpha}^2 (v_\alpha(i) - v_\alpha(j))^2 = d\sum_{\alpha=2}^{n} \frac{1}{(1-\lambda_\alpha)^2}(v_\alpha(i) - v_\alpha(j))^2, \tag{2.18}$$

where $(\lambda_\alpha)_{1\leq\alpha\leq n}$ and $(v_\alpha)_{1\leq\alpha\leq n}$ are respectively the eigenvalues and right eigenvectors of the matrix P.
The symmetrically normalized Laplacian matrix is defined as (see [7])

$$\mathcal{L} = I - D^{-1/2}AD^{-1/2} = D^{1/2}(I-P)D^{-1/2}, \tag{2.19}$$

it follows that $\sigma_\alpha = 1 - \lambda_\alpha$ is an eigenvalue of $\mathcal{L}$ corresponding to the eigenvector $s_\alpha = D^{1/2}v_\alpha$ for all $\alpha = 1, ..., n$. Combining with (2.20), we have

$$r_{ij}^2 = d\sum_{\alpha=2}^{n} \frac{1}{\sigma_\alpha^2}(v_\alpha(i) - v_\alpha(j))^2, \tag{2.20}$$

with $v_\alpha = D^{-1/2}s_\alpha$, where $(\sigma_\alpha)_{1\leq\alpha\leq n}$ and $(s_\alpha)_{1\leq\alpha\leq n}$ are respectively the eigenvalues and right eigenvectors of the matrix $\mathcal{L}$. □

We present some crucial observations that must be taken into account in using random walks to detect community structure. In addition, we give theoretical comparisons between the *simple walk* with our algorithm.

Remark 2.6 *The transition matrix P has real eigenvalues satisfying:*

$$1 = \lambda_1 > \lambda_2 \geq \ldots \geq \lambda_n \geq -1.$$

Remark 2.7 *In [6], let G be a network with apparent community structure. Then, P also has $m-1$ eigenvalues that are approximately 1, where m is the number of well-defined communities.*

Remark 2.8 *In [6], the eigenvectors associated to these first $m-1$ nontrivial eigenvalues, also have a community characteristic, i.e, the elements that correspond to vertices within the same community are roughly the same.*

We restate the result of M. Latapy and P. Pons in [28].

Theorem 2.9 *([28, Theorem 1]) The distance r is related to the spectral properties of the matrix P by:*

$$r_{ij}^2 = \sum_{\alpha=2}^{n} \lambda^{2t}(v_\alpha(i) - v_\alpha(j))^2. \tag{2.21}$$

where $(\lambda_\alpha)_{1\leq\alpha\leq n}$ and $(v_\alpha)_{1\leq\alpha\leq n}$ are respectively the eigenvalues and right eigenvectors of the matrix P.

The following table summaries the results we have obtained in Theorems 2.5 and 2.9.

Walktrap	$r_{ij}^2 = \sum_{\alpha=2}^{n} \lambda_\alpha^{2t}(v_\alpha(i) - v_\alpha(j))^2$	$f_1(\lambda_\alpha) = \lambda_\alpha^{2t}$
Our distance	$r_{ij}^2 = \sum_{\alpha=2}^{n} \frac{1}{(1-\lambda_\alpha)^2}(v_\alpha(i) - v_\alpha(j))^2$	$f_2(\lambda_\alpha) = \frac{1}{(1-\lambda_\alpha)^2}$

It can be seen that all of the above distances have $(v_\alpha(i) - v_\alpha(j))^2$ as their common factor, only differing in terms of $f_1(\lambda_\alpha)$ and $f_2(\lambda_\alpha)$. This fact together with **Remark 4.1**, **Remark 4.2** and **Remark 4.3** show that one need to weaken (ideally, eliminate) the "noisy" eigenvalues, i.e, the ones with absolute values close to zero in order to yield a not good graph partition.

Remark 2.10 *With the Walktrap algorithm, the eigenvalues with absolute value close to* 1 *change much "slower", compared to those who are close to zero when we use the power t of P.*

Remark 2.11 *With our algorithm, dominate eigenvalues go close to* 1 *with very fast speed and the remaining eigenvalues increase at a very slow speed. For example:*

λ_α	*0.95*	*0.9*	*0.6*	*0.5*
$f_1(\lambda_\alpha) = \lambda_\alpha^{2t}$ *with* $t = 3$	$\approx$ *0.735*	$\approx$ *0.531*	$\approx$ *0.047*	$\approx$ *0.016*
$f_2(\lambda_\alpha) = \frac{1}{(1-\lambda_\alpha)^2}$	*400*	*100*	*6.25*	*4*

2.3 Relation with singular value decomposition on digraphs

For a strongly connected digraph G, let $\Phi^{1/2} = diag[\sqrt{\phi_i}]$. Yanhua and Z. L. Zhang [34] defined the normalized digraph Laplacian matrix (Diplacian for short) $\Gamma = [\Gamma_{ij}]$ for the graph G as follows.

Definition 2.12 *([34, Definition 3.2]) The Diplacian Γ is defined as*

$$\Gamma = \Phi^{1/2}(I - P)\Phi^{-1/2}. \tag{2.22}$$

Theorem 2.13 *([34, Theorem 3.4]) Let $G = (V, E, A)$ be a strongly connected digraph with the normalized fundamental matrix $\mathcal{Z} = \Phi^{1/2}Z\Phi^{-1/2}$. Then $\mathcal{Z} = \Gamma^+$ is the pseudoinverse of the Diplacian matrix Γ, with Γ is Diplacian matrix.*

Theorem 2.14 *The distance r is related to the spectral properties of the matrix P by:*

$$r_{ij}^2 = \sum_{\alpha=2}^{n} \frac{1}{{\sigma_\alpha}^2}(w_\alpha(i) - w_\alpha(j))^2, \tag{2.23}$$

where σ_i, v_i be the ith singular value, the corresponding right singular vectors of Γ and $w_\alpha = \Phi^{-1/2}v_\alpha$.

Proof For digraphs we can express Γ_{ij}^+ directly in terms of the singular values and left/right singular vectors of the generally asymmetric Diplacian matrix Γ. Let σ_i, u_i and v_i be the ith singular value and the corresponding left and right singular vectors of Γ arranged in increasing order, where $\|u_i\|_2 = \|v_i\|_2 = 1$, $i = 1, 2, ..., n$. In particular, $0 = \sigma_1 < \sigma_2 \leq \ldots \leq \sigma_n$. Hence $\Gamma = U\Sigma V^T$, where $\Sigma = diag(\sigma_1, ..., \sigma_n)$, $U = (u_1, ...u_n)$, $V = (v_1, ..., v_n)$ with $UU^T = VV^T = I$ and $\mathcal{Z} = \Gamma^+ = V\Sigma^+U^T$, where $\Sigma^+ = (\sigma_1^+, ..., \sigma_n^+)$. Therefore,

$$Z = \sum_{\alpha=1}^{n} \sigma_\alpha^+ \Phi^{-1/2} v_\alpha (\Phi^{1/2}u_\alpha)^T.$$

It follows that

$$Z_{i\bullet} = \sum_{\alpha=1}^{n} \sigma_\alpha^+ \phi_i^{-1/2} v_\alpha(i)\Phi^{1/2}u_\alpha = \Phi^{1/2}\sum_{\alpha=1}^{n} \sigma_\alpha^+ \phi_i^{-1/2} v_\alpha(i)u_\alpha.$$

We put this formula into equation (2.8). Then we use the Pythagorean theorem with the orthonormal family of vectors $(v_\alpha)_{1\leq\alpha\leq n}$, while keeping in mind that the vector v_1 is constant to remove the case $\alpha = 1$ in the sum. Finally we have

$$r_{ij}^2 = \sum_{\alpha=2}^{n} {\sigma_\alpha^+}^2(\phi_i^{-1/2}v_\alpha(i) - \phi_j^{-1/2}v_\alpha(j))^2 = \sum_{\alpha=2}^{n} \frac{1}{{\sigma_\alpha}^2}(w_\alpha(i) - w_\alpha(j))^2, \tag{2.24}$$

where σ_i, v_i be the i-th singular value, the corresponding right singular vectors of Γ and $w_\alpha = \Phi^{-1/2}v_\alpha$. □

Remark 2.15 *We see that the result in directed graphs (2.23) is similar to (2.16) in undirected graphs.*

3 Spectral method for directed graphs

In this section, we propose the spectral method to solve the modularity for the directed graph. More general, our method can be applied for weighted digraphs. We denote those digraph as $G = (V, E, A)$ where $V = \{1, 2, ..., n\}$, $A = (a_{ij})_{i,j=\overline{1,n}}$ is a non-negative matrix and $a_{ij} > 0$ if and only if there is at least one directed edge from vertex i to vertex j. The out-degree of vertex i is defined as $d_i^+ = \sum_{j=1}^n a_{ij}$ while the in-degree of vertex i is computed as $d_i^- = \sum_{j=1}^n a_{ji}$. In the undirected case, because edges do not have directed, $d_i^+ = d_i^- = d_i$, but in the directed case, edges have directed so the property above is not ensured. The volume of the directed graph G is referred as $d = \sum_{i=1}^n d_i + = \sum_{i=1}^n d_i^-$. Because the type of degree we mainly use is out-degree and the out-degree we use in the directed case often has the same meaning as the degree in the

undirected case, we denote $d_i = d_i^+$ and $\boldsymbol{d} = (d_1, d_2, ..., d_n)$, the out-degree vector of the directed G.

We denote the transition probability matrix as $P = D^{-1}A$ where $D = diag(\boldsymbol{d})$ is diagonal matrix and $\boldsymbol{d} = (d_1, d_2, ..., d_n)$.

$$D = diag(\boldsymbol{d}) = \begin{bmatrix} d_1 & 0 & ... & 0 \\ 0 & d_2 & ... & 0 \\ ... & ... & \ddots & ... \\ 0 & 0 & ... & d_n \end{bmatrix}.$$

We set $\{X_k\}_{k=0,1,2,...}$, the finite Markov chain with the transition probability matrix P and the state space $\mathbb{S}$.

If the Markov chain $\{X_k\}$ is irreducible and aperiodic then there exists the limitation of P^k:

$$P_\infty = \lim_{t\to\infty} P^k = \begin{bmatrix} \phi_1 & \phi_2 & ... & \phi_n \\ \phi_1 & \phi_2 & ... & \phi_n \\ ... & ... & \ddots & ... \\ \phi_1 & \phi_2 & ... & \phi_n \end{bmatrix},$$

where $\phi = (\phi_1, \phi_2, ..., \phi_n)$ is the stationary distribution corresponding to P, that means ϕ_i is the limitation of probability the chain reaches to state $i \in \mathbb{S}$:

$$\phi_i = \lim_{k\to\infty} P\left(X_k = i\right).$$

The condition of the directed graph for the random walk to have irreducible property is strongly connected while the aperiodic condition is difficult to satisfy in the real graph, but we can overcome this problem by adding only one self-edge at an arbitrary vertex with small weight. More specially, when we demonstrate ϕ distribution exists, we can compute this vector more simple by finding the left-eigenvector of P^T:

$$\phi P = \phi. \tag{3.1}$$

Similar to the case of undirected graphs, we define modularity Q for directed graphs as follows

$$Q = \sum_{ij} \left(P_{ji}\phi_j - \phi_i\phi_j\right)\delta_{C_iC_j}, \tag{3.2}$$

where P_{ji} is transition propability of random walk process rom j-th vertex to i-th vertex, $\phi = (\phi_1, \phi_2, ..., \phi_n)$ is stationary distribution stationary and

$$\delta_{C_iC_j} = \begin{cases} 1 & C_i = C_j, \\ 0 & otherwise, \end{cases}$$

with C_i is community of i-th vertex.

Our proposed modularity is similar to research in [5, 27]. Their modularity is called as LinkRank and their idea approaches in the direction of using PageRank to determine LinkRank values at positions (i, j) while ours is based entirely on a random walk, using spectral method and

reasoning by that direction for clustering. Their modularity function is shown in the following formula:

$$Q_{linkrank} = \sum_{ij} \left[L_{ij} - \pi_i \pi_j \right] \delta \left(c_i, c_j \right), \tag{3.3}$$

where π_i is the PageRank value of website i and $L_{ij} = \pi_i K_{ij}$ with K is Google matrix [5].

Our problem is how to maximize Q function in the equation (3.2). In this paper, we use the spectral method to solve this problem. As with many modularity methods in the past, we start with the graph including two communities C_1 and C_2.

Firstly, we replacing $\delta_{C_i C_j}$ by simple variable:

$$s_i = \begin{cases} 1 & \text{if } i \in C_1, \\ -1 & \text{if } i \in C_2. \end{cases}$$

Therefore, the modularity function becomes:

$$Q = \frac{1}{2} \sum_{ij} \left(P_{ji} \phi_j - \phi_i \phi_j \right) \left(s_i s_j + 1 \right). \tag{3.4}$$

By defining M, modularity matrix of this modularity version by:

$$M_{ij} = P_{ji} \phi_j - \phi_i \phi_j. \tag{3.5}$$

The Q function will be shown more simple as:

$$Q = \frac{1}{2} \sum_{ij} M_{ij} (s_i s_j + 1). \tag{3.6}$$

Because $\sum_i P_{ji} = 1$ and $\sum_i \phi_i = 1$, we also have

$$\sum_i \sum_j M_{ij} = \sum_j \phi_j \left(\sum_i \left(P_{ji} - \phi_i \right) \right) = \sum_j \phi_j (1 - 1) = 0.$$

Therefore, Q can be shown as the following formula:

$$Q = \frac{1}{2} \sum_{ij} M_{ij} s_i s_j. \tag{3.7}$$

Our original problem is then

$$\max_s Q = \max_s \frac{1}{2} \sum_{ij} M_{ij} s_i s_j,$$

where $s = (s_1, s_{2,}..., s_N)\,; s_i \in \{-1, +1\}$.

Because the dicrete value of s_i, problem is NP-hard problem. We can choose a constraint of the form $\sum_i a_i s_i^2 = \sum_i a_i$ for any set of nonnegative constants a_i. Because $s_i \in \{-1, +1\}$, this constraint is satisfied. The advantage of this relaxing step is making the value space of s_i continuous, which makes the new problem easier to solve. In this paper we will find it convenient to make a special choice, leading to a spectral modularity optimization algorithm that is different in some important respects from previous algorithms. We set $a_i = \phi_i$, the observed degrees of

the vertices, so that our constraint takes the form. We will solve relaxed problem with the value space of s being continuous,

$$\max_s \sum_{ij} M_{ij} s_i s_j, \tag{3.8}$$

where $\sum_i \phi_i s_i^2 = 1$. The result of dividing the graph into communities will be determined based on the sign of each element in solution vector of problem (3.8). Which means i-th vertex will be in C_1 or C_2 depending on the sign of s_i. We apply Lagrange multiplier μ to solve problem (3.8):

$$\mathcal{L}(s, \mu) = -\sum_{ij} M_{ij} s_i s_j + \mu \left(\sum_i \phi_i s_i^2 - 1 \right). \tag{3.9}$$

Then we obtain the following equation about partial derivative of Lagrange function above:

$$-\sum_j M_{ij} s_j + 2\mu \pi_i s_i = 0. \tag{3.10}$$

Let $\lambda = 2\mu$, we obtain:

$$\sum_j M_{ij} s_j = \lambda \phi_i s_i. \tag{3.11}$$

Use the result from above equation, we obtain:

$$Q_{max} = \frac{1}{2} \sum_{ij} M_{ij} s_i s_j = \frac{\lambda}{2} \sum \phi_i s_i^2 = \frac{\lambda}{2}. \tag{3.12}$$

We can express equation (3.11) as matrix notation with $\boldsymbol{s} = (s_1, s_2, ..., s_n)^T \in R^{n \times 1}$:

$$M\boldsymbol{s} = \lambda \Phi \boldsymbol{s}, \tag{3.13}$$

where

$$\Phi = diag(\phi) = \begin{bmatrix} \phi_1 & 0 & ... & 0 \\ 0 & \phi_2 & ... & 0 \\ ... & ... & \ddots & ... \\ 0 & 0 & ... & \phi_n \end{bmatrix}.$$

The solution above can be simplified further. By combining equation (3.11) and definition of modularity matrix M:

$$\sum_j P_{ji} \phi_j s_j = \lambda \phi_i s_i + \sum_j \phi_i \phi_j s_j. \tag{3.14}$$

Equation (3.14) can be shown in matrix notation with $\mathbf{1} = (1, 1, ..., 1)^T \in R^{n \times 1}$:

$$P^T \Phi \boldsymbol{s} = \Phi \left(\lambda \boldsymbol{s} + \left(\boldsymbol{\phi^T s} \right) \mathbf{1} \right). \tag{3.15}$$

We will simplify equation (3.15) with the following transformation:

$$1^T P^T \Phi s = 1^T \Phi \left(\lambda \boldsymbol{s} + \left(\phi^T \boldsymbol{s} \right) \mathbf{1} \right).$$

Since P is transition probability matrix, therefore

$$1^T P^T = \left(\sum_{i=1}^{N} P_{1i}, \sum_{i=1}^{N} P_{2i}, ..., \sum_{i=1}^{N} P_{ni} \right) = (1, 1, ..., 1) = 1^T,$$

we obtain:

$$(\lambda - 1)\, \phi^T \boldsymbol{s} = -\phi^T \left(\phi^T \boldsymbol{s} \right) \mathbf{1},$$

and $\phi^T \mathbf{1} = \mathbf{1}$, we have:

$$\lambda \phi^T \boldsymbol{s} = 0.$$

Going back to formula (3.11), we can see that if $\lambda = 0$ there will be a solution $\boldsymbol{s} = \mathbf{1}$, which means all signs of elements are positive and all vertices have the same community. It is not useful when obtain that result. So the condition of λ is $\lambda \neq 0$. Then we obtain:

$$\phi^T \boldsymbol{s} = 0.$$

Therefore, the equation (3.15) can be rewrote more simple :

$$P^T \Phi \boldsymbol{s} = \lambda \Phi \boldsymbol{s}. \tag{3.16}$$

Multiply on the left of both sides of the equation (3.16) by $\Phi^{-1/2}$ and let $u = \Phi^{1/2} s$, we have:

$$\Phi^{-1/2} P^T \Phi^{1/2} u = \lambda u. \tag{3.17}$$

Let $L = \Phi^{1/2} P \Phi^{-1/2}$, we have:

$$L^T u = \lambda u. \tag{3.18}$$

It is clear that $\boldsymbol{s} = \mathbf{1}$ and $\lambda = 1$ is one pair of eigenvalue and eigenvector of equation (3.18). But as we have explained before, $\boldsymbol{s} = \mathbf{1}$ does not help to detect communities. On the other hand, by using the Perron-Frobenius, $\lambda = 1$ is the maximum eigenvalue. We need to find the second most positive eigenvalue then determine its corresponding eigenvector, then using the sign of each element to determine the community each vertex should belong to. We note that, though it is not all, most networks have a positive eigenvalue smaller than 1, and we will assume this for our case.

Remark 3.1 *In addition to being used for clustering, our modularity Q can also be used to evaluate clustering quality. It can be generalized for partitioning a network into c communities*

$$Q = \sum_{ij} \left[P_{ji} \phi_j - \phi_i \phi_j \right] \delta_{C_i C_j}, \tag{3.19}$$

where C_i is the community membership of vertex i, C_j is the community membership of vertex j, and $\delta_{C_i C_j}$ equals 1 if i and j belong to the same community, otherwise it equals 0.

4 Algorithms

4.1 Algorithms using hitting times

Algorithm for undirected graphs: We apply the same Walktrap algorithm (see [28]) with the presented modifications in the distance formula, we will use our distance in (2.1). We note

that, for an undirected graph G, we have $\phi_i = \frac{d_i}{d}$ where d_i is the degree of vertex i and d is the total of degrees of graph G. Thus we only need to find the hitting time. To calculate hitting time, there are many algorithms to quickly calculate hitting times, such as Algorithm 3 in [15].
Algorithm for directed graphs: We apply the same Walktrap algorithm (see [28]) with the presented modifications in the distance formula, we will use our distance in (2.1) and we need to replace modularity Q introduced in [24, 26], which relies on the fraction of edges e_C inside community C and the fraction of edges a_C bound to community C:

$$Q(\mathcal{P}) = \sum_{C \in \mathcal{P}} e_C - a_C^2, \tag{4.1}$$

by modularity Q introduced in [1] for directed graphs

$$Q = \frac{1}{2m} \sum_{i,j} \left(A_{ij} - \frac{d_i^{out} d_j^{in}}{m} \right) \delta_{C_i C_2}. \tag{4.2}$$

Here $A_{ij} = 1$ if there is a link between vertices i and j, d_i^{out} is the out-degree of vertex i, d_i^{in} is the in-degree of vertex i, C_i is the community membership of vertex i, C_j is the community membership of vertex j, and $\delta_{C_i C_2} = 1$ if i and j belong to the same community, otherwise it equals 0. Or replace modularity Q in (4.1) by our modularity Q in (3.19):

$$Q = \sum_{ij} \left[P_{ji} \phi_j - \phi_i \phi_j \right] \delta_{C_i C_j}. \tag{4.3}$$

Thus we only need to find the hitting times and the stationary distribution. To calculate the stationary distribution and hitting times we use Algorithm 1 and Algorithm 4 in [10] with directed graph with n vertices and m edges. It was showed how to compute each quantity in time $\tilde{O}(m^{3/4}n + mn^{2/3})$, where the $\tilde{O}$ notation suppresses polylogarithmic factors in n. We note that real networks are sparse graphs, so we have a computational complexity of $\tilde{O}(n^{7/4})$.

4.2 Algorithm using spectral

From the results of Section 3. We have to divide a graph into two clusters, we only need to find the eigenvector $U_I = (u_{1I}, u_{2I}, ..., u_{nI})$ corresponding to the eigenvalue w_I, where w_I is the largest real eigenvalue other than 1. Then, we split the cluster as follows: If u_{iI} has a positive sign, then vertex i belongs to cluster 1, and if u_{iI} has a negative sign, then vertex i belongs to cluster 2. Moreover, our proposed method for modularity function can be summarized as the following pseudocode of Algorithm 1 and more specially, we also integrate the mechanism to detect more than 2 communities through the iterative algorithm. The stopping condition of this iterative algorithm is all potential community be considered to divide. One special point in our mechanism is the additional condition about increasing the value of modularity of accepting division from arbitrary community.

5 Examples

Evaluating a community detection algorithm is a difficult task because one needs some test graphs whose community structure is already known. A classical approach is to use randomly

Algorithm 1 Spectral method for directed graphs

Input: Directed graph $G = (V, E), |V| = N$.

Output: Set of communities $R = \{C_1, C_2, ..., C_K\}$.

Initial $R = \{C_{ori}\}$ with $C_{ori} = \{v_1, v_2, ..., v_N\}$ $(v_i \in V \forall i)$
Intial dictionary $c = \{C_{ori} : True\}$
$Z = \{C \in R | \, c(C) = True\}$
while $|Z| > 0$ **do**
    Choose $C_{temp} \in R$, $c(C_{temp}) = True$
    while $c(C_{temp}) = True$ **do**
        $G_{temp} = (C_{temp}, E_{temp})$ is the vertex-induced sub graph of G
        Ensure G_{temp} strongly connected property and period is equal to 1
        Approximate $\phi = (\phi_1, \phi_2, ..., \phi_N)$
        Modularity of $R = Q_1$
        Compute $L = \Phi^{1/2} P \Phi^{-1/2}$
        Set $U = \begin{bmatrix} U_1 & U_2 & ... & U_N \end{bmatrix} \in R^{N \times N}$ $\big(U_i \in R^{N \times 1}$ is $i-$th eigenvector$\big)$ and
        $W = (w_1, w_2, ..., w_N)$ (w_i is $i-$th eigenvalue)
        if $\nexists j \in \{1, 2, ..., N\} : \, w_j \in R, \, w_j \neq 1$ **then**
            $c(C_{temp})$=False
            $Z = \{C \in R | \, c(C) = True\}$
            break
        end if
        $w_I = \max_{w_i} \{w_i, i \in \{1, 2, ..., N\} | \, w_i \in R, w_i \neq 1\}$. Choose $U_I = (u_{iI})_{i=1,2,...,N}$
        $C_1 = \{v_i \in V_{temp} | \, u_{iI} > 0\}, C_2 = \{v_i \in V_{temp} | \, u_{iI} < 0\}$.
        if $|C_1| \cdot |C_2| = 0$ **then**
            $c(C_{temp})$=False.
            $Z = \{C \in R | \, c(C) = True\}$
            break
        end if
        $R_{temp} = R \cup \{C_1, C_2\} \backslash \{C_{temp}\}$
        Modularity of $R_{temp} = Q_2$
        if $Q_1 < Q_2$ **then**
            $R \leftarrow R_{temp}$
            $c(C_{temp})$ =False, $c(C_1)$ =True, $c(C_2)$ =True
            $Z = \{C \in R | \, c(C) = True\}$
        else
            $c(C_{temp})$=False.
            $Z = \{C \in R | \, c(C) = True\}$
        end if
    end while
end while
return R

generated graphs with communities. Here we will use this approach and generate the graphs as follows.

Planted l-partition model: The first generator model we introduce is the planted l-partition

model [17]. By determining the number of groups l, the number of vertex each group g, two probability of inter-cluster p_{in} and intra-cluster p_{out}, we obtain one random graph with some property:

- The average degree of one vertex is $E[k] = p_{in}(g-1) + p_{out}g(l-1)$.
- All communities have same size.
- All vertices have approximately the same degree because each community can be seen as one random graph proposed by Erdos and Renyi. In those random graphs, each pair of vertices is connected with equal probability p_{in} independently of other pairs.

Gaussian random partition generator: The next generator graph model is the Gaussian random partition generator [17], which overcomes partly disadvantage of planted $l-$partition model above about degree distribution of vertices. Different from planted $l-$partition model, the size of the community in Gaussian random partition generator is a random variable of Gaussian distribution. The parameters need to determined of Gaussian random partition generator are:

- Number of vertices in graph: N;
- Mean of community's size: m and variance of community's size: σ;
- Edge probability of inter p_{in} and intra-cluster p_{out}.

LFR benchmark graph: The final model to random graph we show is the LFR benchmark graph [17], which is stronger than Gaussian random partition generator in creating the difference between the size of the community. The LFR graph is created with the assumption that the degree of vertex and the size of the community is a random variable of power distribution. LFR is also interested in edge density in and out of each community as each vertex will have a certain proportion of edges in its community and outside the community. The main parameters of the LFR graph:

- The exponents parameter of degree's vertex is τ_1 and τ_2 is the corresponding exponents parameter of size's community.
- The density proportion of edges of each vertex between its community and others community is $\frac{1-\mu}{\mu}$

In the Example part, we have tested the proposed methods for community detection of an undirected and a directed graph using hitting times. We also use the spectral method to detect the community with the our modularity function. Most of the results we get are significant because they closely matching to the label of the graph.

After clustering a graph, we need to evaluate the quality of that clustering. Therefore, next we will present two metrics to evaluate the clustering quality.

Rand Index

To evaluate the quality of the partition found by the algorithms, we compare them to the original generated partition. To achieve this, we use the Rand index corrected by Hubert and Arabie [29, 18] which evaluates the similarities between two partitions. Since the graph we used to test is built from the parameters, so we have the labels of the communities, where the performance

testing of the proposed algorithm will be more accurate. Rand Index of 2 graph clustering results $\mathcal{P}_1 = \left\{C_1^1.C_2^1, ...\right\}, \mathcal{P}_2 = \left\{C_1^2.C_2^2, ...\right\}$ is calculated according to the following formula: $RandIndex(\mathcal{P}_1, \mathcal{P}_2) \in (0, 1]$:

$$\text{RandIndex } (\mathcal{P}_1, \mathcal{P}_2) = \frac{N^2 \sum_{ij} |C_i^1 \cap C_j^2|^2 - \sum_i |C_i^1|^2 \sum_j |C_j^2|^2}{\frac{1}{2}N^2 \left(\sum_i |C_i^1|^2 + \sum_j |C_j^2|^2\right) - \sum_i |C_i^1|^2 \sum_j |C_j^2|^2}. \tag{5.1}$$

Jaccard Index

Another metric that also measures the label-based effectiveness of the community from the randomly generated graph is the Jaccard Index [19, 33]. The Jaccard index of two sets A and B is measured as the following formula, $\text{Jaccard}\,(A, B) \in [0, 1]$:

$$\text{Jaccard } (A, B) = \frac{|A \cap B|}{|A \cup B|}. \tag{5.2}$$

We have two clustering results for a graph G are

$$\mathcal{H} = \{\mathcal{H}_1, \mathcal{H}_2, ..., \mathcal{H}_h\} \text{ and } \mathcal{K} = \{\mathcal{K}_1, \mathcal{K}_2, ..., \mathcal{K}_k\},$$

we measure the Jaccard index of each pair $(\mathcal{H}_i, \mathcal{K}_j)_{i \in \overline{1,h}, k \in \overline{1,k}}$ and then plot the heatmap based on the matrix J where $J_{ij} = \text{Jaccard}\,(\mathcal{H}_i, \mathcal{K}_j)$. For a more concise representation of the performance based on the Jaccard Index, we compute average the Jaccard index of the pairs of most similar clusters for each row

$$MJC = \frac{1}{h} \sum_{i=1}^{h} \max_j J_{ij} = \frac{1}{h} \sum_{i=1}^{h} \max_j \text{Jaccard}\,(\mathcal{H}_i, \mathcal{K}_j). \tag{5.3}$$

5.1 Examples for algorithm using hitting times

Examples for undirected graphs

Example 5.1 *We create a graph G_1 according to the Planted l-partition model with the following parameters: number of communities $l = 8$, size of each community $g = 30$, probability of edge appearing in each community $p_{in} = 0.5$ and probability of edge outside the community $p_{out} = 0.01$, then we get the graph G_1 as shown in Figure 1. We will apply the algorithm using hitting times for undirected graphs to cluster this graph.*

We get the following result:
• Figure 1 is an illustration of label of the random partition graph of Example 5.1.
• Figure 2 is an illustration of the community detection results of Example 5.1.
• Figure 3 shows the matrix J where $J_{ij} = \text{Jaccard}\,(\mathcal{H}_i, \mathcal{K}_j)$ with $\mathcal{H} = \{\mathcal{H}_1, \mathcal{H}_2, ..., \mathcal{H}_h\}$ is clustering in the way of creating graph G_1 and $\mathcal{K} = \{\mathcal{K}_1, \mathcal{K}_2, ..., \mathcal{K}_k\}$ is the clustering result of graph G_1 when apply the algorithm using hitting times for undirected graphs.
• Table 1 is a table of results measured by the previously introduced metrics.

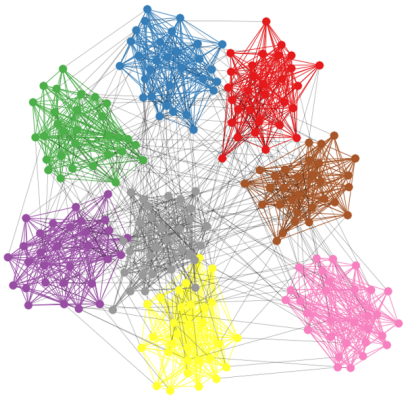

Figure 1: Graph G_1

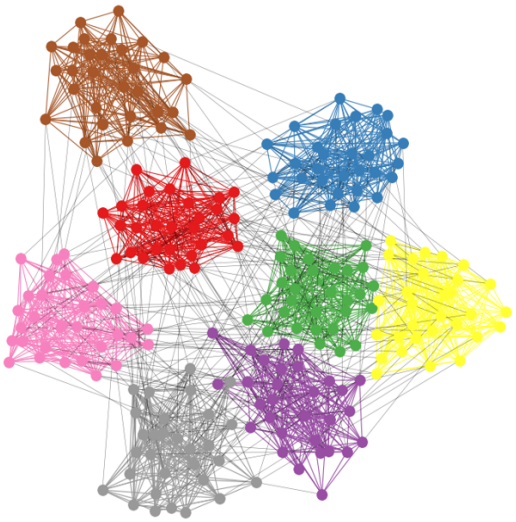

Figure 2: Illustration of the clustering results of G_1 when applying the algorithm using hitting times for undirected graphs

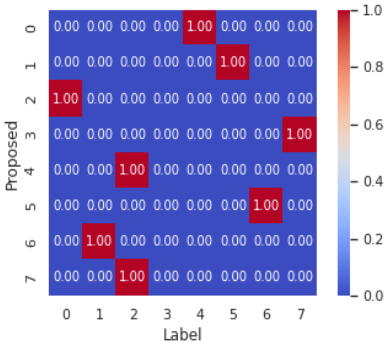

Figure 3: Heatmap of Jaccard

Example 5.2 *We create a graph G_2 according to a Gaussian random partition generator model with the following parameters: number of vertices $N = 200$, average size of the community $m = 40$, variance $\sigma = 20$, probability of edge appearing in each community $p_{in} = 0.5$ and the edge probability outside the community $p_{out} = 0.01$, then we get the graph G_2 as shown in Figure 4. We will apply the algorithm using hitting times for undirected graphs to cluster this graph.*

We get the following result:
• Figure 4 is an illustration of label of the random partition graph of Example 5.2.
• Figure 5 is an illustration of the community detection results of Example 5.2.
• Figure 6 shows the matrix J where $J_{ij} = \text{Jaccard}\,(\mathcal{H}_i, \mathcal{K}_j)$ with $\mathcal{H} = \{\mathcal{H}_1, \mathcal{H}_2, ..., \mathcal{H}_h\}$ is clustering in the way of creating graph G_2 and $\mathcal{K} = \{\mathcal{K}_1, \mathcal{K}_2, ..., \mathcal{K}_k\}$ is the clustering result of

graph G_2 when apply the algorithm using hitting times for undirected graphs.
• Table 1 is a table of results measured by the previously introduced metrics.

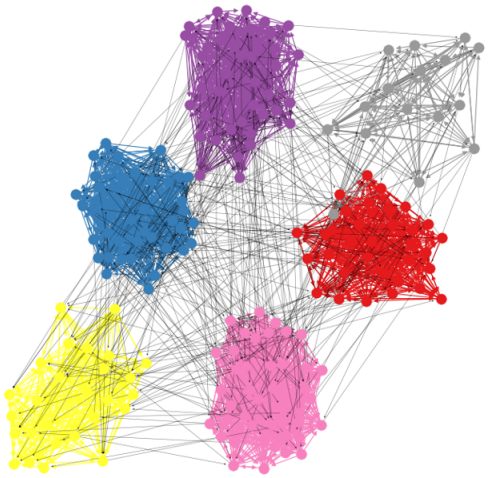

Figure 4: Graph G_2

Figure 5: Illustration of the clustering results of G_2 when applying the algorithm using hitting times for undirected graphs

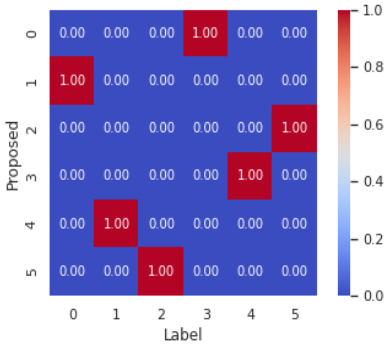

Figure 6: Heatmap of Jaccard

Type of graph	$G = (V, E)$	Q of label	Q of our result	RandIndex	Mean of Jaccard
Planted l-partition model	$G_1 = (240, 2211)$	0.733	0.733	1	1
Gaussian random partition generator	$G_2 = (200, 3956)$	0.721	0.721	1	1

Table 1: Result of proposed method with hitting times for undirected graph.

Remark 5.3 *According to the results in Table 1, we see that Rand Index and Mean of Jaccard are both equal to* 1*, so the clustering results using our algorithm using hitting times for undirected graphs completely coincide with the clusters generated by both graph generation Planted l-partition model and Gaussian random partition generator.*

Example 5.4 *We create a graph according to the LFR standard graph model with the following parameters: the number of vertices* $N = 100$*, the parameter of the exponential distribution corresponding to the order of the vertex* $\tau_1 = 1$*, the parameter of the distribution exponential to the size of the community* $\tau_2 = 2$*, the ratio of the outer edges of the community* $\mu = 0.1$*, the mean value and the maximum value of the degree of the vertex are 5 and 10, the minimum and the maximum, about the size of the community is 10 and 25, then we get the graph* G_3 *as shown in Figure 7.*
a. We will apply the algorithm using hitting times for graph G_3*.*
b. We will apply the Walktrap algorithm graph G_3*.*

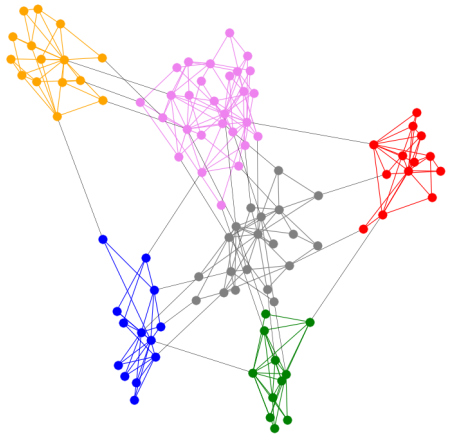

Figure 7: Graph G_3

a. Applying the algorithm using hitting times for graph G_3, we obtain clustering results as shown in Figure 8 and Heatmap of Jaccard as shown in Figure 9. We will apply the algorithm using hitting times for graph G_3. b. Applying the Walktrap algorithm for graph G_3, we obtain

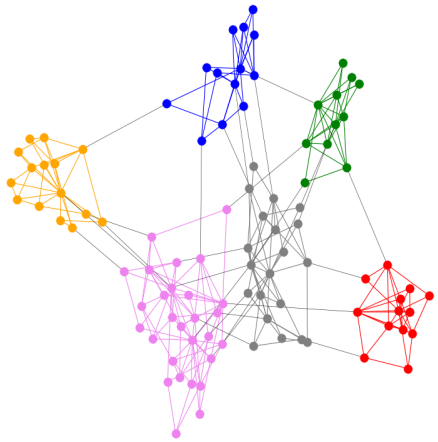

Figure 8: Result of proposed algorithm with hitting time

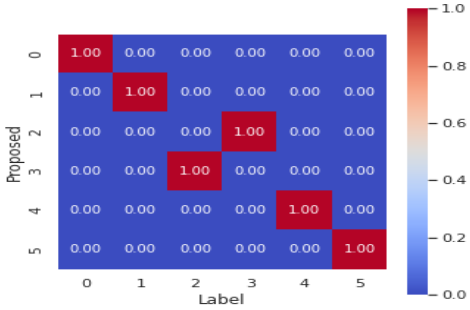

Figure 9: Heatmap of Jaccard

clustering results as shown in Figure 10 and Heatmap of Jaccard as shown in Figure 11. We will apply the algorithm using hitting times for graph G_3.

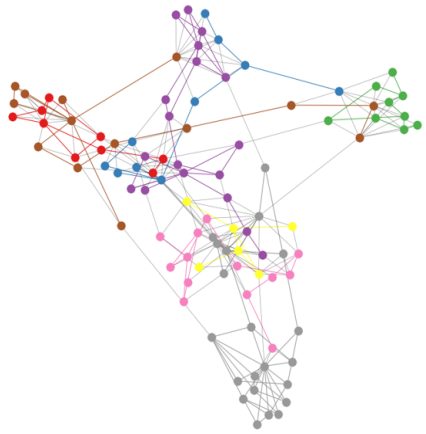

Figure 10: Result of Walktrap algorithm

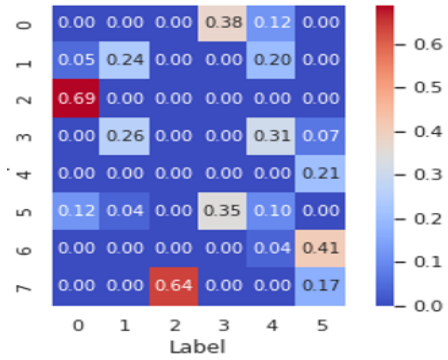

Figure 11: Heatmap of Jaccard

Remark 5.5 *In some cases, our algorithm using hitting times is more efficient than the Walktrap algorithm.*

Examples for directed graphs

Example 5.6 *We create a graph G_4 according to the Planted l-partition model with the following parameters: number of communities $l = 6$, size of each community $g = 25$, probability of edge appearing in each community $p_{in} = 0.5$ and probability of edge outside the community $p_{out} = 0.01$, then we get the graph G_4 as shown in Figure 12.*
a. We will apply the algorithm using hitting times for directed graphs with modularity Q in 4.2

to cluster this graph.
b. We will apply the algorithm using hitting times for directed graphs with our modularity Q in 4.3 to cluster this graph.

a. Using modularity Q in 4.2 we get the following result:
• Figure 12 is an illustration of label of the random partition graph of Example 5.6.
• Figure 13 is an illustration of the community detection results of Example 5.6.
• Figure 14 shows the matrix J where $J_{ij} = \text{Jaccard}\,(\mathcal{H}_i, \mathcal{K}_j)$ with $\mathcal{H} = \{\mathcal{H}_1, \mathcal{H}_2, ..., \mathcal{H}_h\}$ is clustering in the way of creating graph G_4 and $\mathcal{K} = \{\mathcal{K}_1, \mathcal{K}_2, ..., \mathcal{K}_k\}$ is the clustering result of graph G_4 when apply the algorithm using hitting times for directed graphs.
• Furthermore, we also have a modularity function test that is recommended to be used to find the best number of clusters, and the results are obtained quite effectively as some other quantities such as directed modularity. Figure 15 is an illustration after surveying the modularity value of the partition per iteration, the maximum value is corresponding to the best partition.
• Table 2 is a table of results measured by the previously introduced metrics.

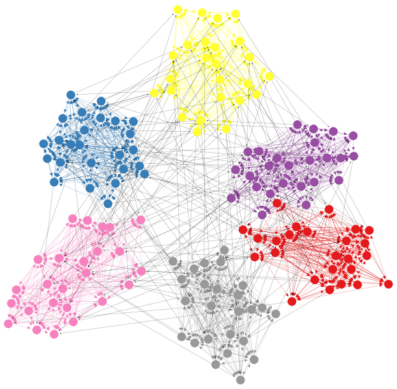

Figure 12: Graph G_4

Figure 13: Illustration of the clustering results of G_4 when applying the algorithm using hitting times for directed graphs

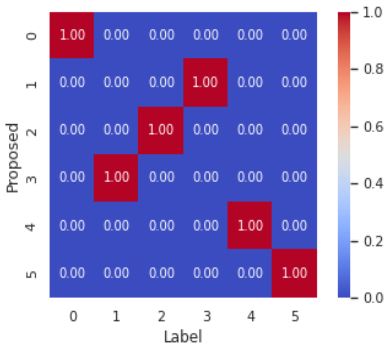

Figure 14: Heatmap of Jaccard

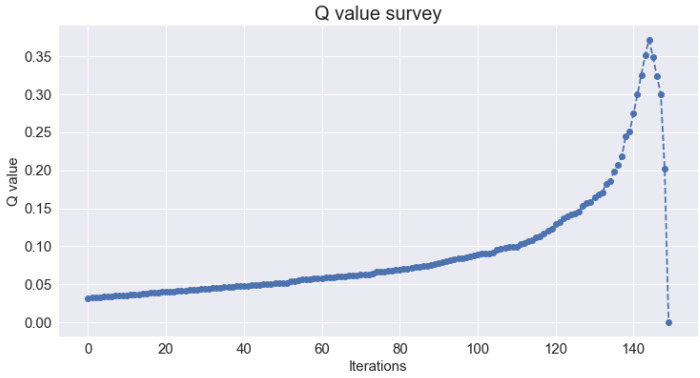

Figure 15: Choose the best number of communities based on normal modularity function of example 5.4.

b. Using our modularity Q in 4.3 we get the following result:
We also use our modularity to determine the best partition in the set of partitions which are obtained after using the Walktrap algorithm with our proposed distance. The proposed modularity function also brings a good result when its results are the same as the label of the random partition graph. Because the role of proposed modularity is to determine the number of communities and its result is as good as normal modularity, quality metrics as rand index, mean of jaccard is same as normal modularity results. The Figure 16 illustrates after surveying the modularity value based on our modularity of partition per iteration.

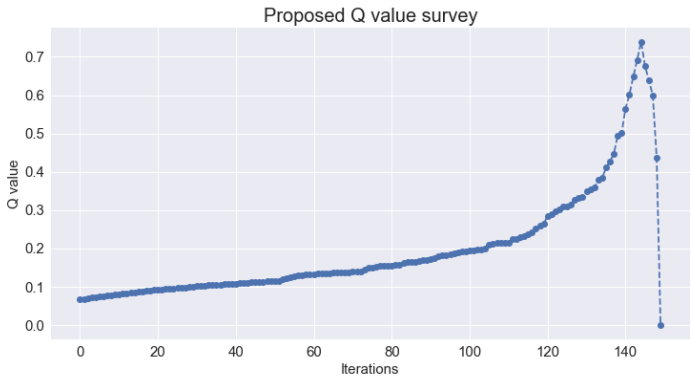

Figure 16: Choose the best number of communities based on our modularity function of example 5.4.

Remark 5.7 *Because the result after performing our proposed algorithm is the same as the label of the random partition graph, the modularity value based on both normal modularity and proposed modularity of both label and our clustering results are identical.*
Therefore, in Table 2, we only show one modularity value of each kind of modularity function.

Example 5.8 *We create a graph G_5 according to a Gaussian random partition generator model with the following parameters: number of vertices $N = 260$, average size of the community $m = 50$, variance $\sigma = 20$, probability of edge appearing in each community $p_{in} = 0.5$ and the edge probability outside the community $p_{out} = 0.01$, then we get the graph G_5 as shown in Figure 17.*
a. We will apply the algorithm using hitting times for directed graphs with normal modularity Q in 4.2 to cluster this graph.
b. We will apply the algorithm using hitting times for directed graphs with our modularity Q in 4.3 to cluster this graph.

a. Using modularity Q in 4.2 we get the following result:
• Figure 17 is an illustration of label of the random partition graph of Example 5.8.
• Figure 18 is an illustration of the community detection results of Example 5.8.
• Figure 19 shows the matrix J where $J_{ij} = \text{Jaccard}\,(\mathcal{H}_i, \mathcal{K}_j)$ with $\mathcal{H} = \{\mathcal{H}_1, \mathcal{H}_2, ..., \mathcal{H}_h\}$ is clustering in the way of creating graph G_5 and $\mathcal{K} = \{\mathcal{K}_1, \mathcal{K}_2, ..., \mathcal{K}_k\}$ is the clustering result of graph G_5 when apply the algorithm using hitting times for directed graphs.
• Figure 20 is an illustration after surveying the modularity value of the partition per iteration, the maximum value is corresponding to the best partition.
• Table 2 is a table of results measured by the previously introduced metrics.

Figure 17: Graph G_5

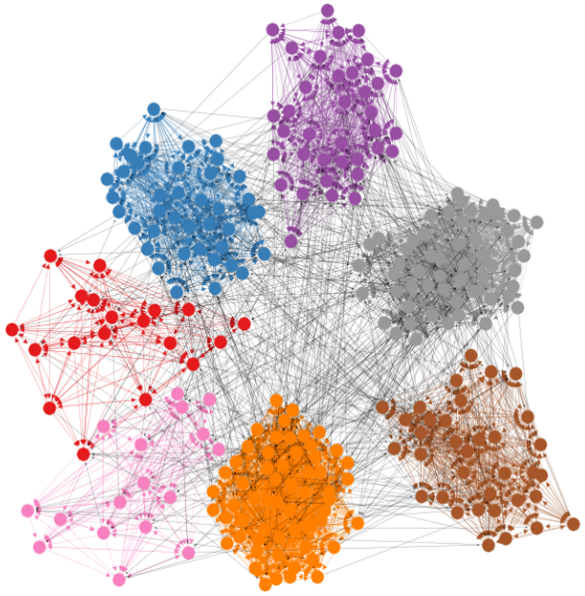

Figure 18: Illustration of the clustering results of G_5 when applying the algorithm using hitting times for directed graphs

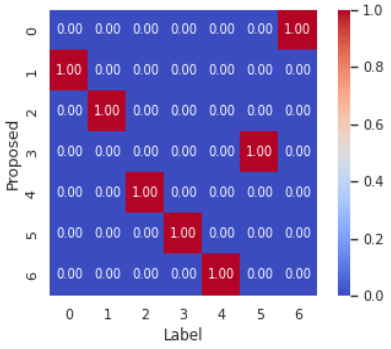

Figure 19: Heatmap of Jaccard

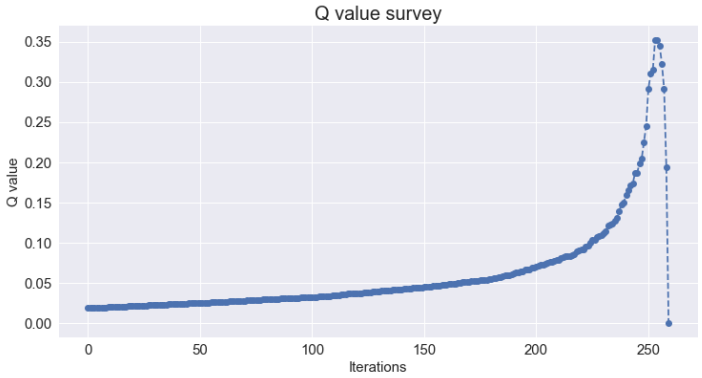

Figure 20: Choose the best number of communities based on normal modularity function of example 5.5.

b. Using our modularity Q in 4.3 we get the following result:
As the previous example, results of using our modularity is the same as the label of G_5. The following figure 21 as illustration after surveying the the modularity value based on our modularity of the partition per iteration.

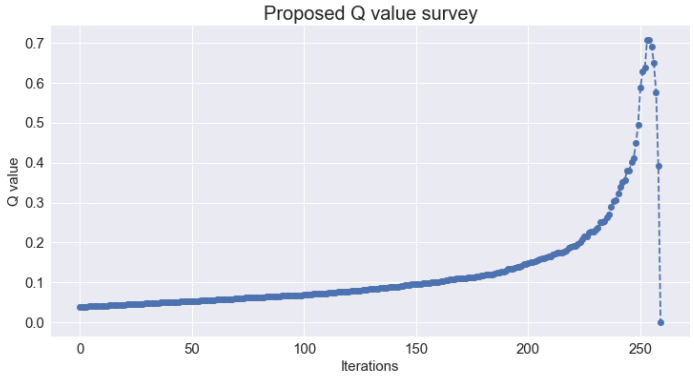

Figure 21: Choose the best number of communities based on our modularity function of example 5.6.

Type of graph	$G = (V, E)$	Normal Q of label	Normal Q of our result	RandIndex	Mean of Jaccard
Planted l-partition model	$G_3 = (150, 2116)$	0.372	0.739	1	1
Gaussian random partition generator	$G_4 = (260, 6265)$	0.352	0.708	1	1

Table 2: Result of proposed method with hitting times for directed graph.

Remark 5.9 *According to the results in Table 2, we see that Rand Index and Mean of Jaccard are both equal to* 1*, so the clustering results using our algorithm using hitting times for directed graphs completely coincide with the clusters generated by both graph generation Planted l-partition model and Gaussian random partition generator.*

5.2 Examples for spectral method

Example 5.10 *We create a graph* G_6 *according to the Planted l-partition model with the following parameters: number of communities* $l = 6$*, size of each community* $g = 25$*, probability of edge appearing in each community* $p_{in} = 0.5$ *and probability of edge outside the community* $p_{out} = 0.01$*, then we get the graph* G_6 *as shown in Figure 22. We will apply spectral method for directed graphs to cluster this graph.*

We get the following result:
• Figure 22 is an illustration of label of the random partition graph of Example 5.10.
• Figure 23 is an illustration of the community detection results of Example 5.10.
• Figure 24 shows the matrix J where $J_{ij} = \text{Jaccard}\,(\mathcal{H}_i, \mathcal{K}_j)$ with $\mathcal{H} = \{\mathcal{H}_1, \mathcal{H}_2, ..., \mathcal{H}_h\}$ is clustering in the way of creating graph G_6 and $\mathcal{K} = \{\mathcal{K}_1, \mathcal{K}_2, ..., \mathcal{K}_k\}$ is the clustering result of graph G_6 when apply the spectral method for directed graphs.
• Table 3 is a table of results measured by the previously introduced metrics.

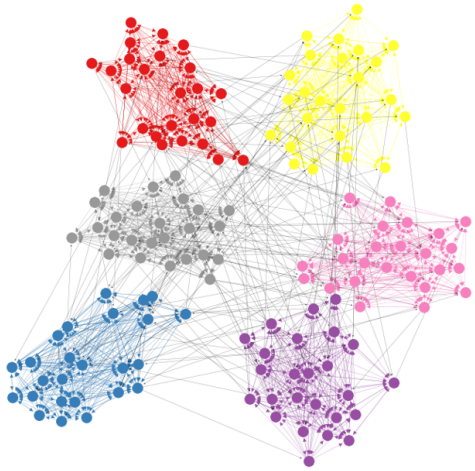

Figure 22: Graph G_6

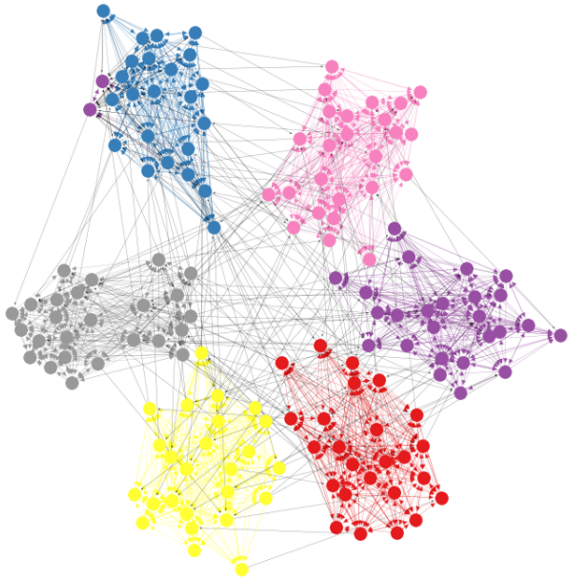

Figure 23: Illustration of the clustering results of G_6 when applying the spectral method for directed graphs

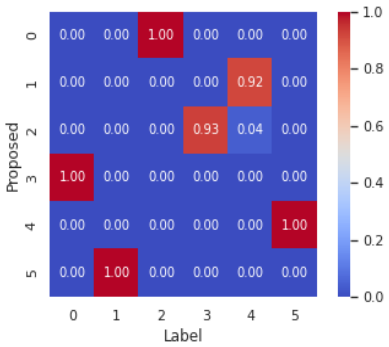

Figure 24: Heatmap of Jaccard

Type of graph	$G = (V, E)$	Normal Q of label	Normal Q of our result	RandIndex	Mean of Jaccard
Planted l-partition model	$G_5 = (150, 2123)$	0.373	0.363	0.969	0.976
Gaussian random partition generator	$G_6 = (200, 4604)$	0.352	0.342	0.967	0.966

Table 3: Result of proposed method with spectral method for directed graph.

Example 5.11 *We create a graph G_7 according to a Gaussian random partition generator model with the following parameters: number of vertices $N = 200$, average size of the community $m = 40$, variance $\sigma = 16$, probability of edge appearing in each community $p_{in} = 0.5$ and the edge probability outside the community $p_{out} = 0.01$, then we get the graph G_7 as shown in Figure 25. We will apply spectral method for directed graphs to cluster this graph.*

We get the following result:
- Figure 25 is an illustration of label of the random partition graph of Example 5.11.
- Figure 26 is an illustration of the community detection results of Example 5.11.
- Figure 27 shows the matrix J where $J_{ij} = \text{Jaccard}\,(\mathcal{H}_i, \mathcal{K}_j)$ with $\mathcal{H} = \{\mathcal{H}_1, \mathcal{H}_2, ..., \mathcal{H}_h\}$ is

clustering in the way of creating graph G_7 and $\mathcal{K} = \{\mathcal{K}_1, \mathcal{K}_2, ..., \mathcal{K}_k\}$ is the clustering result of graph G_7 when apply the spectral method for directed graphs.
• Table 3 is a table of results measured by the previously introduced metrics.

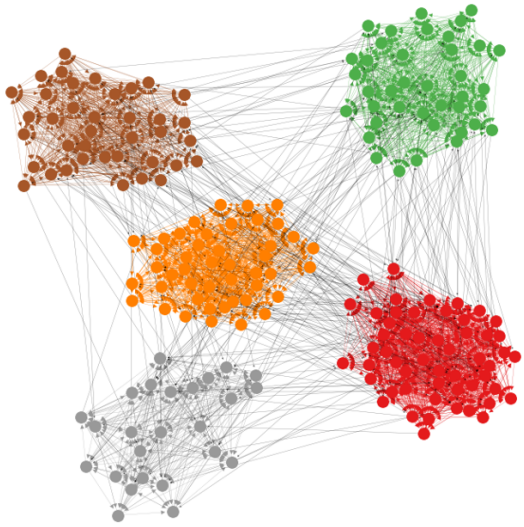

Figure 25: Graph G_7

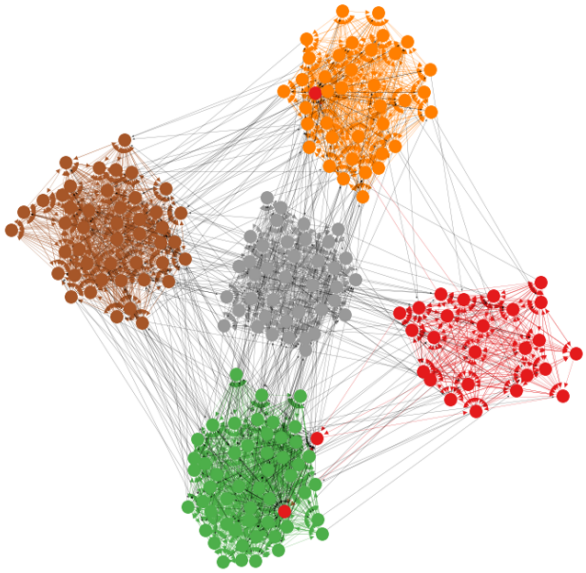

Figure 26: Illustration of the clustering results of G_7 when applying the spectral method for directed graphs

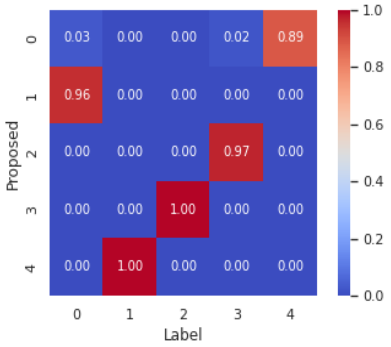

Figure 27: Heatmap of Jaccard

Remark 5.12 *According to the results in Table 3, we see that Rand Index and Mean of Jaccard are both very close to* 1*, so the clustering results using our spectral method for directed graphs*

almost coincide with the clusters generated by both graph generation Planted l-partition model and Gaussian random partition generator.

6 Conclusion and discussion

This paper not only proposed a new definition of distance between vertices based on hitting times and stationary distribution of random walk process on the graph but also proposed to use the spectral method to optimize the new modularity function. Our formula for calculating the distance between two vertices is based on our assumption that if two vertices i and j belong to the same community, their transitions to other vertices are quite similar. In this paper, we take a close look at the random walk process on both directed and undirected graphs based on hitting time values and stationary distribution. After proposing a new definition of distance, we use a mechanism similar to the Walktrap algorithm to perform clustering. Details of the meaningful results we have achieved:

- Proposing a new distance formula, which shows more clearly the relationship between the vertices through the random walk process. This definition is for both undirected and directed graphs.
- Using a similar mechanism as the Walktrap algorithm with the new distance formula.
- Proposing to use the spectral method for a new modularity function.
- Creating a mechanism to automatically divide more than two clusters by an iterative algorithm when using the spectral method with the stopping condition is not able to improve the modularity value more.
- Apply the proposed models on random partition graphs and get significant results.
- Our proposed method can be applied to both the undirected graph and the directed graph.
- Extend the Walktrap method (which was applied to undirected graphs) in a natural way on directed graphs, and our model is not dependent on any symmetrization method.

In the future, deep intervention into the processes occurring in the graph will yield a lot of hidden information about the relationship between the vertices. Our proposed distance formula or the modularity function using the spectral method can be extended to many other clustering mechanisms.

Acknowledgments

This research was supported by the International Center for Research and Postgraduate Training in Mathematics, Project code: ICRTM04-2021.01.

References

[1] A. Arenas, J. Duch, A. Fernández and S. Gómez, Size reduction of complex networks preserving modularity. New Journal of Physics, 9 (6), 176, 2007.

[2] M. S. Aldenderfer and R. K. Blashfield. Cluster Analysis. Number 07-044 in Sage University Paper Series on Quantitative Applications in the Social Sciences. Sage, Beverly Hills, 1984.

[3] J. Bagrow, E. Bollt. A local method for detecting communities. Physical Review E 72(4 Pt 2). 2005: 046108.

[4] P. Bremaud. Markov Chains: Gibbs Fields, Monte Carlo Simulation and Queues (Texts in Applied Mathematics, 31) 2nd ed. 2020 Edition.

[5] S. Brin, L. Page, The anatomy of a large-scale hypertextual web search engine, Comput. Netw. ISDN Syst. 30 (1-7) (1998) 107–117

[6] A. Capoccia, V.D.P. Servedioa, G. Caldarellia, F. Colaiori, Detecting communities in large networks, July 2005 Physica A: Statistical Mechanics and its Applications 352(2-4):669-676.

[7] F. Chung, Spectral Graph Theory. American Mathematical Society. ISBN 978-0821803158. (1997) [1992].

[8] F. Chung, Laplacians and the cheeger inequality for directed graphs, Annals of Combinatorics 9 (2005) 1–19.

[9] L. da F. Costa. Hub-based community flnding, arXiv:cond-mat/0405022, 2004.

[10] M. B. Cohen, J. Kelner, J. Peebles, R. Peng, A. Sidford, A. Vladu, Faster Algorithms for Computing the Stationary Distribution, Simulating Random Walks, and More, Annual IEEE Symposium on Foundations of Computer Science, 2016,Page(s):583 - 592.

[11] A. Clauset. Finding local community structure in networks. Physical Review E, 72:026132, 2005.

[12] A. Clauset, M. E. J. Newman, and C. Moore. Finding community structure in very large networks. Physical Review E, 70(6): 066111, 2004

[13] S. v. Dongen. Graph Clustering by Flow Simulation. PhD thesis, University of Utrecht, May 2000.

[14] N. Dugué, A. Perez. Directed Louvain: maximizing modularity in directed networks. [Research Report] Université d'Orléans. 2015.

[15] L. Enzhi and L. Zhengyi, Frustrated random walks: A faster algorithm to evaluate node distances on connected and undirected graphs, Physical Review E 102, 052135, 2020.

[16] B. S. Everitt, S. Landau, and M. Leese. Cluster Analysis. Hodder Arnold, London, 4th edition, 2001.

[17] S. Fortunato, "Community detection in graphs," Physics Reports, vol. 486, pp. 75–174, 2010.

[18] L. Hubert and P. Arabie. Comparing partitions. Journal of Classification, 2:193–218, 1985.

[19] P. Jaccard, The Distribution of the Flora in the Alpine Zone, The New Phytologist Vol. 11, No. 2, 37-50, 1912.

[20] B. S. Khan and M. A. Niazi, "Network Community Detection: A Review and Visual Survey," 2017.

[21] D. Lai, H. Lu, C. Nardini, Finding communities in directed networks by pagerank random walk induced network embed- ding, Physica A 389 (2010) 2443–2454.

[22] E. A. Leicht and M. E. J. Newman, Community structure in directed networks, Physical Review Letters, 100, 118703 (2008).

[23] C. Moore. The Computer Science and Physics of Community Detection: Landscapes, Phase Transitions, and Hardness. Bull. EATCS 121, 2017.

[24] M. E. J. Newman. Fast algorithm for detecting community structure in networks. Physical Review E, 69(6):066133, 2004.

[25] M. E. J. Newman, Spectral methods for network community detection and graph partitioning, Phys. Rev. E, vol. 88, p. 042822, October 2013.

[26] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. Physical Review E, 69(2), p. 026113, 2004.

[27] L. Page, S. Brin, R. Motwani, T. Winograd, The pagerank citation ranking: Bringing order to the web, in: WWW '98: Proceedings of the 7th International World Wide Web Conference, 1998, pp. 161–172.

[28] P. Pons and M. Latapy. Computing communities in large networks using random walks, Journal of Graph Algorithms and Applications, volume 10. no. 2, 2006, Pages 191–218, 2006.

[29] W.M. Rand. Objective criteria for the evaluation of clustering methods. Journal of the American Statistical association, 66:846–850, 1971.

[30] J. Reichardt, S. Bornholdt. Detecting fuzzy community structures in complex networks with a potts model. Physical Review Letters, 93:218701, 2004.

[31] V. Satuluri, S. Parthasarathy, Symmetrizations for clustering directed graphs, in: EDBT '11: Proceedings of the 14th International Conference on Extending Database Technology, 2011, pp. 343–354.

[32] C. Takacs. On the fundamental matrix of finite state Markov chains, its eigensystem and its relation to hitting times. Math. Pannon., 17(2):183–193, 2006.

[33] TT. Tanimoto, An Elementary Mathematical theory of Classification and Prediction. Internal IBM Technical Report. 1958.

[34] L. Yanhua and Z. L. Zhang, Digraph Laplacian and the Degree of Asymmetry, nternet Mathematics Vol. 8, No. 4: 381–401, 2012.