Community Detection in Directed Graphs via Hitting Times and K-Means Clustering

Do Duy Hieu[a], Bui Quoc[a], and Nguyen Hai Tuan[b]

[a]International Centre of Research and Postgraduate Training in Mathematics, Institute of Mathematics, Vietnam Academy of Science and Technology
[b]Institute of Mathematics, Vietnam Academy of Science and Technology

Email: ddhieu@math.ac.vn (Do Duy Hieu), quoccon1999@gmail.com (Bui Quoc), haituann@gmail.com(Nguyen Hai Tuan).

Abstract

Community detection is a critical problem in network science, with applications in sociology, biology, and computer science. Extending methods from undirected to directed graphs remains challenging due to their inherent asymmetry. This paper introduces the Directed Euclidean Commute Time (DECT), a novel distance metric based on hitting times of random walks. We integrate this metric with the K-Means algorithm to propose the K-Means DECT algorithm for directed graphs. We achieve efficient computation of DECT by expressing the DECT distance through the Singular Value Decomposition (SVD) of the Diplacian matrix. Finally, experiments on real-world networks validate the effectiveness of our proposed approach.

1 Introduction

Community detection in networks is a fundamental problem with widespread applications in sociology, biology, and computer science [10]. Identifying communities within a network allows for a better understanding of the underlying structure and interactions between entities.

A widely adopted and well-regarded approach for graph clustering involves embedding the vertices of a graph into a vector space $\mathbb{R}^n$. This embedding enables the definition of distances between vertices and facilitates the application of traditional clustering algorithms developed for vector spaces. Among these algorithms, K-Means has been particularly popular and effective in such scenarios, as demonstrated in studies like [7, 6]. This method leverages the geometric representation of graph data to simplify the complex structure of graphs, making it more suitable for established clustering techniques.

Among these methods, we are particularly interested in leveraging random walks for community detection. Defining distances between vertices in a graph using random walks has recently become a growing focus. For instance, the Walktrap algorithm [18] utilizes random walks to identify communities, while [24] investigates hitting times as a foundation for defining distances in undirected graphs.

In previous work by Dang, Do, and Phan [7], we utilized hitting times of random walks to define distance metrics for both undirected and directed graphs. This approach demonstrated

the potential of using random walk properties to effectively capture the structural relationships within graphs. Building upon this foundation, in this paper, we further develop the idea of leveraging hitting times from random walks to define a novel distance metric. To enhance computational efficiency, we utilize algebraic methods to establish a connection between this distance metric and the singular value decomposition (SVD) of the Diplacian matrix. This relationship allows for faster and more scalable computations. Based on this metric, we introduce the K-Means DECT algorithm for community detection in directed graphs. Finally, we validate the effectiveness of our method through extensive experiments on real-world graphs, comparing its performance against several existing algorithms.

2 Distance in Directed Graphs Using Hitting Times

2.1 Distance Definition

This section extends the previously developed algorithm for undirected graphs [24] to directed graphs. In a directed graph G, the relationship between two nodes i and j can be characterized by the time it takes for a random walker to travel between them. When i and j belong to the same community, the average round-trip time (from i to j and back) is typically small. Conversely, this travel time increases significantly if they belong to different communities. Based on this observation, we define the distance measure.

Definition 2.1 *The Directed Euclidean Commute Time (DECT) distance on a directed graph is a metric that quantifies the expected number of steps for a random walker to travel from node i to node j and then return to node i. It is formally defined as:*

$$\delta(i,j) = \sqrt{C_{ij}},$$

where

$$C_{ij} = H_{ij} + H_{ji},$$

and H_{ij} is the expected hitting time from node i to node j, while H_{ji} is the expected hitting time from node j back to node i.

2.2 Calculating DECT Distance Using Singular Value Decomposition

Calculating distances using hitting times, despite the availability of fast computational methods, remains a challenging task. However, the corresponding graph representation matrices are often sparse, allowing for more efficient computations. Therefore, this section establishes a relationship between distances based on hitting times and the Singular Value Decomposition (SVD) of the directed graph Laplacian matrix (Dilaplacian). This approach allows us to indirectly compute distances by performing SVD on the Dilaplacian matrix. Before presenting this relationship, we introduce some necessary results as a foundation.

For a strongly connected digraph G, let $\Phi^{1/2} = diag[\sqrt{\phi_i}]$. Yanhua and Z. L. Zhang [13] defined the normalized digraph Laplacian matrix (Diplacian for short) $\Gamma = [\Gamma_{ij}]$ for the graph G as follows.

Definition 2.2 *([13]) The Diplacian Γ is defined as*

$$\Gamma = \Phi^{1/2}(I - P)\Phi^{-1/2}. \tag{2.1}$$

Where $P = D^{-1}A$ is the transition matrix of graph G and $\Phi^{1/2}$ defined as above.

To establish a relationship between the normalized fundamental matrix and the Diplacian matrix, we refer to the result by Yanhua and Z. L. Zhang [13]. They showed that the normalized fundamental matrix, derived from the stationary distribution and the fundamental matrix, is the pseudoinverse of the Diplacian matrix. This crucial connection allows us to compute the normalized fundamental matrix directly through the components of the Diplacian matrix. The result is formally stated as follows:

Theorem 2.3 *([13]) 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 as the Diplacian matrix defined above.*

This section presents the results of the expression of H_{ij} through the fundamental matrix Z and the stationary distribution Φ. First, we have the following theorem:

Theorem 2.4 (Takacs, [22]) *The expected hitting time matrix H is defined as:*

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

The results below, based on [22] and [13], allow for the calculation of the hitting time H_{ij} and the Directed Euclidean Commute Time (DECT) $C_{ij} = H_{ij} + H_{ji}$ using the fundamental matrix Z and the Diplacian matrix. Using the relationship between the Diplacian matrix Γ, its pseudoinverse Γ^+, and the normalized fundamental matrix $\mathcal{Z}$, we can compute the hitting time and DECT by Γ^+, or equivalently, by singular vectors and singular values of Γ.

From Theorem 2.4 and the formula $C_{ij} = H_{ij} + H_{ji}$, we have:

$$C_{ij} = \frac{z_{jj} - z_{ij}}{\phi_j} + \frac{z_{ii} - z_{ji}}{\phi_i}. \tag{2.2}$$

Using $Z = \Phi^{-1/2} \mathcal{Z} \Phi^{1/2} = \Phi^{-1/2} \Gamma^+ \Phi^{1/2}$, the formula 2.2, and Theorem 2.3, we can directly compute the hitting time and DECT via the components of Γ^+, as shown in the following theorem:

Theorem 2.5 ([13]) *The hitting time and DECT for a random walk on a strongly connected directed graph can be computed through the Diplacian pseudoinverse Γ^+ as follows:*

$$C_{ij} = H_{ij} + H_{ji} = \frac{\Gamma^+_{jj}}{\phi_j} + \frac{\Gamma^+_{ii}}{\phi_i} - \frac{\Gamma^+_{ij}}{\sqrt{\phi_i \phi_j}} - \frac{\Gamma^+_{ji}}{\sqrt{\phi_i \phi_j}},$$

where Γ^+_{ij} is the (i, j) element of the matrix Γ^+, and ϕ_i is the stationary distribution of node i.

From this theorem, we derive the following result:

Theorem 2.6 *The Euclidean commute time distance can be computed using the singular values and singular vectors of the matrix Γ as:*

$$C_{ij} = \sum_{k>1} \frac{1}{\sigma_k} \left(\frac{u_{kj}}{\sqrt{\phi_j}} - \frac{u_{ki}}{\sqrt{\phi_i}} \right) \left(\frac{v_{kj}}{\sqrt{\phi_j}} - \frac{v_{ki}}{\sqrt{\phi_i}} \right). \tag{2.3}$$

Here, σ_i is the ith singular value, v_i and u_i are the right and left singular vectors corresponding to the singular value σ_i of the matrix Γ.

Proof For directed graphs, we can expand Γ_{ij}^{+} directly through the left and right singular vectors of the Diplacian matrix Γ. Let σ_i, u_i, and v_i represent the ith singular value and the corresponding left and right singular vectors of Γ, ordered in ascending sequence, where $\|u_i\|_2 = 1$ and $\|v_i\|_2 = 1$ for $i = 1, 2, \ldots, n$. Specifically, we have $0 = \sigma_1 < \sigma_2 \leq \cdots \leq \sigma_n$. Thus, $\Gamma = U\Sigma V^T$, where $\Sigma = \operatorname{diag}[\sigma_i], U = [u_1, \ldots, u_n], V = [v_1, \ldots, v_n]$, and $UU^T = I, VV^T = I$. Therefore, $\Gamma^{+} = V\Sigma^{+}U^T$, where $\Sigma^{+} = \operatorname{diag}\left[\sigma_i^{+}\right]$. Thus, $\Gamma_{ij}^{+} = \sum_{k>1} \frac{1}{\sigma_k} v_{ki} u_{kj}$. Substituting into equation 2.2, we can expand the hitting time and DECT through the singular vectors and singular values of Γ as follows:

$$C_{ij} = \sum_{k>1} \frac{1}{\sigma_k} \left(\frac{v_{kj}u_{kj}}{\phi_j} + \frac{v_{ki}u_{ki}}{\phi_i} - \frac{v_{ki}u_{kj}}{\sqrt{\phi_i\phi_j}} - \frac{v_{kj}u_{ki}}{\sqrt{\phi_j\phi_i}} \right) \tag{2.4}$$

$$= \sum_{k>1} \frac{1}{\sigma_k} \left[\frac{v_{kj}}{\sqrt{\phi_j}} \left(\frac{u_{kj}}{\sqrt{\phi_j}} - \frac{u_{ki}}{\sqrt{\phi_i}} \right) + \frac{v_{ki}}{\sqrt{\phi_i}} \left(\frac{u_{ki}}{\sqrt{\phi_i}} - \frac{u_{kj}}{\sqrt{\phi_j}} \right) \right] \tag{2.5}$$

$$= \sum_{k>1} \frac{1}{\sigma_k} \left(\frac{u_{kj}}{\sqrt{\phi_j}} - \frac{u_{ki}}{\sqrt{\phi_i}} \right) \left(\frac{v_{kj}}{\sqrt{\phi_j}} - \frac{v_{ki}}{\sqrt{\phi_i}} \right). \tag{2.6}$$

□

3 K-Means Algorithm Using DECT Distance

This section introduces the K-Means DECT algorithm, developed based on the original K-Means algorithm. Typically, the cluster centroid is updated by taking the average of all node coordinates in the cluster. However, with DECT distance, we only define distances between nodes without specifying their coordinates. Therefore, we must find a node that best approximates the "centroid." This node is the one for which the sum of squared distances to all other nodes in the cluster is minimized.

The DECT matrix $\mathbf{\Delta}$ is defined with elements $[\mathbf{\Delta}]_{ij} = \delta^2(i,j) = C_{ij}$, representing the squared distance between nodes i and j. For a given number of clusters c, the centroid of each cluster C_l is denoted as $\mathbf{p}_l$. The distance from a node k to cluster C_l is defined as the distance to the centroid of C_l:

$$\mathrm{d}\left(k, C_l\right) = \delta\left(k, \mathbf{p}_l\right).$$

The intra-cluster variance for cluster C_l is given by:

$$J_l = \sum_{k \in C_l} \mathrm{d}^2\left(k, C_l\right).$$

As mentioned, the algorithm updates the cluster centroid by selecting a node that minimizes the sum of squared distances within the cluster.

The objective function to be minimized is the total intra-cluster variance J, which is the sum of the variances J_l of all clusters:

$$J = \sum_{l=1}^{c} J_l = \sum_{l=1}^{c} \sum_{k \in C_l} \mathrm{d}^2\left(k, C_l\right).$$

The algorithm works by assigning nodes to the cluster with the nearest centroid and updating the centroid to minimize the total distance from all nodes in the cluster to the new centroid. Finding the global minimum of J is an NP-hard problem, so most algorithms can only find a local minimum.

Building on these foundations, we introduce the K-Means DECT algorithm specifically designed for directed graphs, detailed below:

Algorithm 1: K-Means DECT Algorithm for Directed Graphs

Input: Directed graph $G = (V, E)$, number of clusters c.

Output: Cluster assignments for nodes in G.

Initialization:

- Select c initial centroids $\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_c$ (e.g., randomly or using a heuristic).
- Compute the DECT matrix, where each element represents the squared distance between nodes in G.

while *The algorithm has not converged* **do**

Clustering Step:

for *each node* $k \in V$ **do**

Assign k to the cluster with the nearest centroid:

$$l = \arg\min_j \delta^2(k, \mathbf{p}_j)$$

Centroid Update Step:

for *each cluster* C_l **do**

Update the centroid $\mathbf{p}_l$ by selecting the node that minimizes the intra-cluster variance:

$$\mathbf{p}_l = \arg\min_{j \in C_l} \sum_{k \in C_l} \delta^2(k, j)$$

Convergence Check:

- Stop if cluster assignments do not change between consecutive iterations.
- Alternatively, stop if the change in centroids is below a predefined threshold:

$$\Delta = \max_l \delta(\mathbf{p}_l^{\text{old}}, \mathbf{p}_l^{\text{new}})$$

where $\Delta \leq \epsilon$, and ϵ is a small tolerance value.

The DECT matrix calculation for all node pairs has a complexity of $O(n^2)$, requiring computing distances between every pair of nodes. In the clustering step, each node k is assigned to the nearest cluster by calculating distances to k centroids, which has a complexity of $O(k)$ per node. With n nodes, this step has a complexity of $O(kn)$ per iteration. The centroid update step involves computing the sum of squared distances between nodes within each cluster. In the worst case, where all nodes belong to a single cluster, this step has a complexity of $O(n^2)$ per iteration. Since $O(n^2)$ from the centroid update step dominates $O(kn)$ from the clustering step, the overall complexity per iteration is $O(n^2)$. Assuming the algorithm converges in T iterations,

the total computational complexity is $O(T \cdot n^2)$. Thus, the **worst-case complexity of the algorithm is** $O(T \cdot n^2)$, where T represents the number of iterations required for convergence.

4 Experiments

4.1 Modularity

We need metrics for datasets with unknown community structures to evaluate whether the algorithms' clustering results are effective. One of the most commonly used metrics is the Modularity function.

The Modularity function is often used to evaluate the quality of clustering. There are various definitions of modularity, but the most relevant is the definition proposed in [4] for directed graphs. Specifically, given a clustering $\mathcal{P} = \{C_1, C_2, ..., C_k\}$ of graph G, where $C(i)$ denotes the cluster label to which node i belongs, Modularity Q_d is defined as:

$$Q_d = \frac{1}{2m} \sum_{i,j} \left[A_{ij} - \frac{d_i^{out} d_j^{in}}{m} \right] \delta_{C_i C_j}, \tag{4.1}$$

where $\delta_{C_i C_j}$ is the Kronecker delta function:

$$\delta_{C_i C_j} = \begin{cases} 1, & \text{if } i = j, \\ 0, & \text{if } i \neq j. \end{cases}$$

A_{ij} is the number of edges from node i to node j, d_i^{out} is the out-degree of node i, d_j^{in} is the in-degree of node j, and m is the total number of edges in the graph.

4.2 Experiments on Real-World Networks

This section introduces and experiments with some commonly used real-world networks. For simplicity, the number of nodes $|V|$ and edges $|E|$ in each network G is denoted $G = (|V|, |E|)$. We shall present our experiment results in groups of 4 graphs in ascending order of size.

K-Means is one of the most widely recognized and utilized algorithms for clustering based on distances between vertices. Several algorithms extend the traditional K-Means by integrating Singular Value Decomposition (SVD), such as oPCA, rPCA, DScore, and DScoreq [6]. In this section, we compare our proposed K-Means DECT algorithm (introduced in Section 3) with these algorithms using real networks of various sizes.

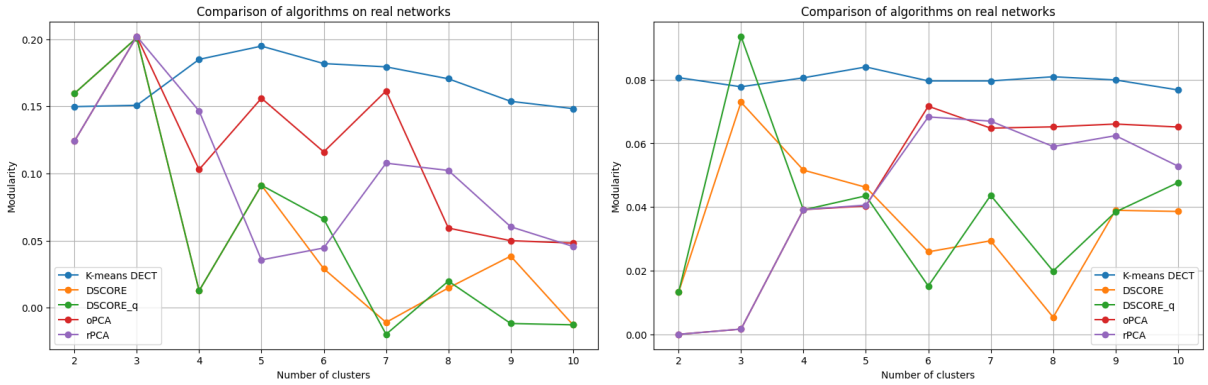
For the comparison, we calculate the directed modularity Q_d (defined in formula 4.1) for the clustering results obtained from each algorithm. To visualize the performance, we will plot a separate graph for each experiment, with the number of clusters (k) on the horizontal axis and the corresponding modularity values on the vertical axis, where k ranges from 2 to 10. Algorithms producing higher modularity values, represented by curves positioned higher on the graph, perform better. This approach allows us to assess and compare the clustering quality of K-Means DECT with the baseline algorithms across small, medium, and large-scale datasets.

Experiments on Small-Scale Networks

We begin by evaluating our algorithm on small networks with fewer than 1000 nodes and 5000 edges. These datasets include various types of networks, such as social, governmental, and biological systems:

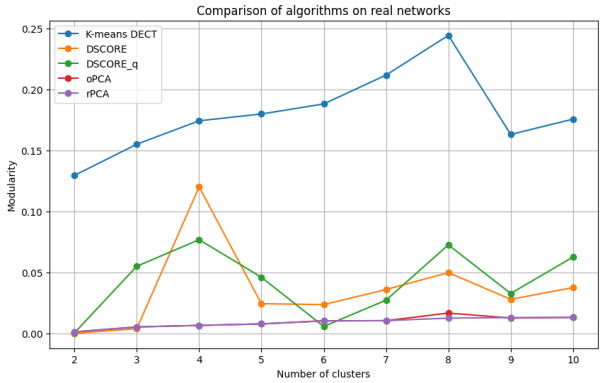
- *Papuan gift-giving (1970)* [20]: This network, denoted as G_1 ($G_1 = (22, 78)$), represents gift-giving relationships (taro exchange) between households in a Papuan village. The data was collected through surveys around 1970.
- *Celegans neural network* [23]: This network, denoted as G_2 ($G_2 = (297, 2359)$), represents the neural connections of the Caenorhabditis elegans nematode.
- *E. coli transcriptional regulatory network* [21]: This network, denoted as G_3 ($G_3 = (424, 577)$), models operons and their pairwise interactions through transcription factor-based regulation within Escherichia coli bacteria.
- *US government agency websites (2018)-Connecticut* [12]: This network, denoted as G_4 ($G_4 = (685, 3201)$), represents the web-based links among government agency websites in Connecticut. Each node corresponds to an agency website, and a directed edge (i, j) indicates the presence of a hyperlink from any webpage on the website i to some webpage on the website j. Data was collected using a crawler. Nodes are annotated with metadata, including the number of pages, website name (related to its government function), URL, and the year it was indexed by the Internet Archive (used as a proxy for the website's creation date). Edge weights represent the number of hyperlinks between websites.

The results are displayed in Figure 1.

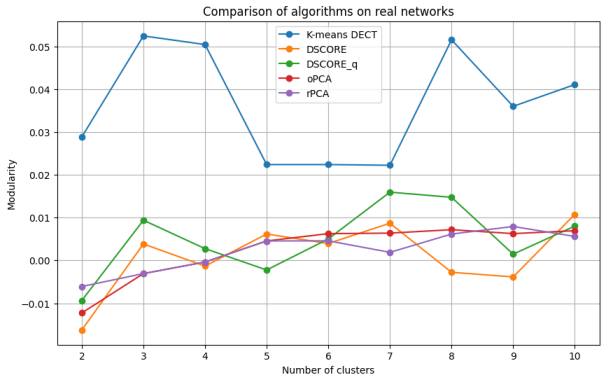


(a) Experiment on the Moreno-Taro network

(b) Experiment on the Neuron network



(c) Experiment on the Ecoli-transcription network



(d) Experiment on the US Agency-Connecticut network

Figure 1: Results of the K-Means DECT algorithm on real-world networks

Remark 4.1 *The K-Means DECT algorithm performs well in detecting communities in small-scale networks, achieving high modularity and apparent clustering. Notably, the method effectively handles diverse types of networks, including social networks (G_1), biological networks (G_2G_3), and governmental information networks (G_4). To be more concrete, for the graph, G_3 and G_4, the K-Means DECT becomes utterly superior to other algorithms. The situation is entirely similar to the case of table G_1 and G_2, except for a few small values of k being subtracted.*

Experiments on Medium-Scale Networks

Next, we apply the K-Means DECT algorithm to medium-scale networks, with nodes ranging from 1000 to 1500 and edges between 2600 and 20000. The datasets include diverse examples, such as governmental, political, and air traffic networks:

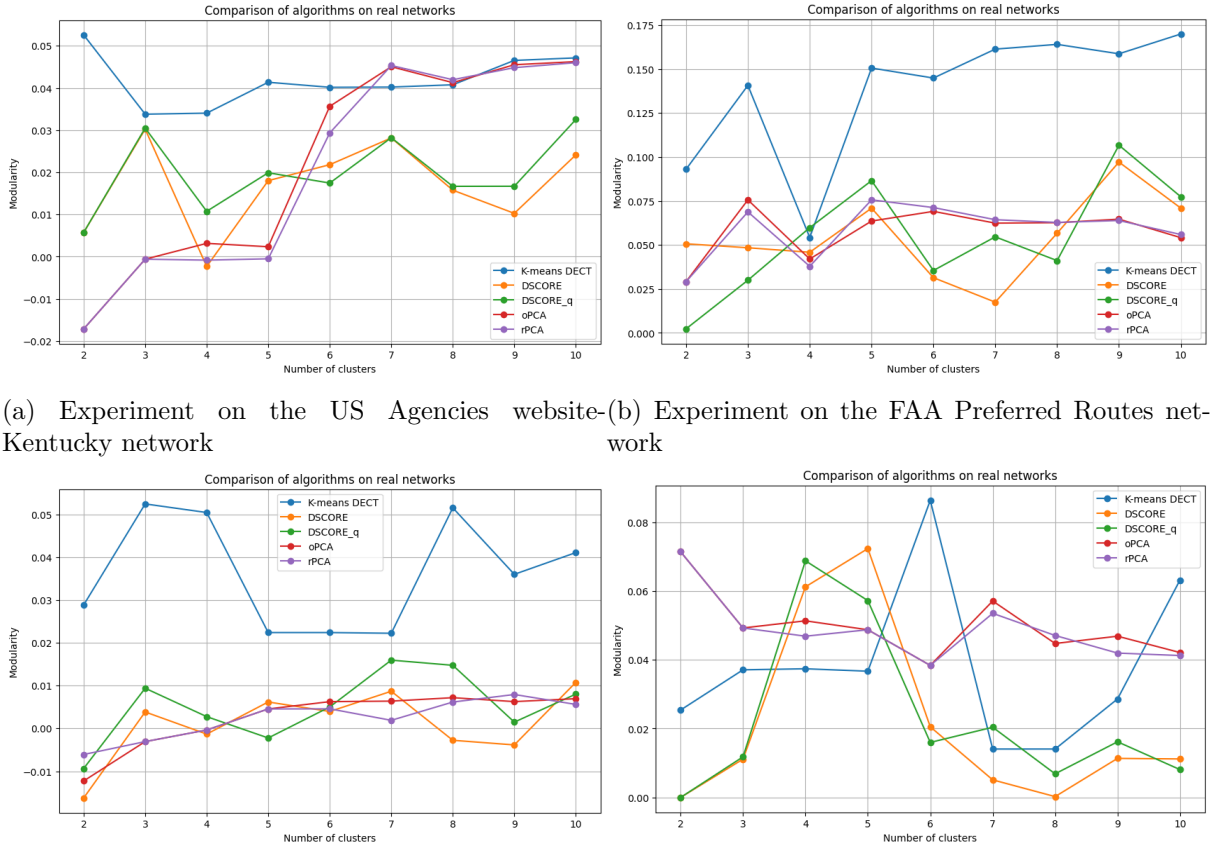
- *US Agency-Kentucky network* [12]: This network, denoted as G_5 ($G_5 = (1049, 7918)$), represents the web-based connections among government agency websites in Kentucky.
- *FAA Preferred Routes (2010)* [3]: This network, denoted as G_6 ($G_6 = (1226, 2615)$), models air traffic routes extracted from the FAA (Federal Aviation Administration) National Flight Data Center (NFDC) preferred routes database (www.fly.faa.gov) before 2010.

Nodes represent airports or service centers, and a directed edge (i, j) indicates a preferred route from airport i to airport j.

- *U.S. government agency websites (2018)-Alabama* [12]: This network, denoted as G_7 ($G_7 = (1281, 5479)$), represents the hyperlink structure among government agency websites in Alabama.

- *Political blogs network (2004)* [2]: This network, denoted as G_8 ($G_8 = (1490, 19090)$), captures the hyperlink structure among US political blogs collected before the 2004 election. Nodes represent individual blogs, and edges indicate hyperlinks, with metadata including each blog's political affiliation.

The results are displayed in Figure 2.



(a) Experiment on the US Agencies website-Kentucky network

(b) Experiment on the FAA Preferred Routes network

(c) Experiment on the US Agencies website-Alabama network

(d) Experiment on the Political blogs network

Figure 2: Results of the K-Means DECT algorithm on real-world networks

Remark 4.2 *In the case of graph G_6 and G_7, Algorithm K-Means DECT outperforms all other algorithms entirely. In the case of graph G_5, the K-Means DECT still has the greatest modularity with $k \leq 5$. However, algorithm K-Means DECT has slightly lower modularity than the oPCA and rPCA algorithms when k exceeds 5. The opposite scenario occurs with the graph G_8 where*

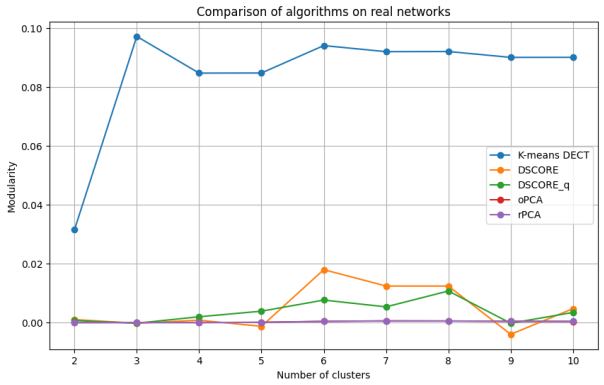
algorithm K-Means DECT outperforms for values of k from 6 onwards but is not particularly effective when k is less than 6.

Experiments on Large-Scale Networks

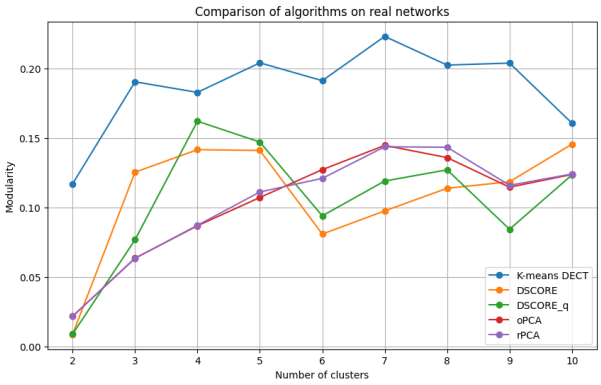
Our analysis then shifts to large-scale networks of 6000 to 8000 nodes and 18000 to 52000 edges. These datasets span software dependencies, user trust relationships, and genetic interactions:

- *Software dependencies (2010)* [25]: This network, denoted as G_9 ($G_9 = (6126, 138706)$), represents software dependencies. Nodes correspond to libraries; a directed edge indicates that one library depends on another.
- *Advogato trust network* [14]: This network, denoted as G_{10} ($G_{10} = (6541, 51127)$), captures trust relationships among users on Advogato, an open-source software community. A directed edge (i, j) indicates that user i trusts user j.
- *Multiplex genetic interactions (2014)* [8]: This network, denoted as G_{11} ($G_{11} = (6980, 18655)$), represents multiplex genetic interactions for Arabidopsis. Layers include (i) physical, (ii) association, (iii) co-localization, (iv) direct, (v) suppressive, and (vi) additive or synthetic genetic interactions. A directed edge (i, j) indicates an interaction from gene i to gene j.
- *Multiplex genetic interactions (2014)* [8]: This network, denoted as G_{12} ($G_{12} = (7747, 19843)$), represents multiplex genetic interactions for Mus. The edge directions have the same interpretation as those in G_{11}.

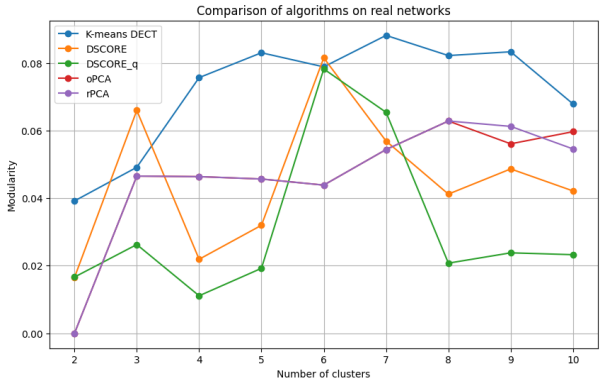
The results are displayed in Figure 3.



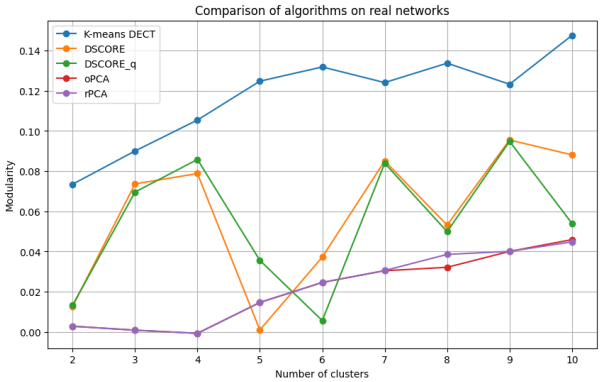
(a) Experiment on the software dependencies network



(b) Experiment on the Advogato trust network.



(c) Experiment on the multiplex genetic interactions network of Mus



(d) Experiment on the multiplex genetic interactions network of Arabidopsis

Figure 3: Results of the K-Means DECT algorithm on real-world networks

Remark 4.3 *In the case of Large networks, the modularity value of K-Means DECT outperforms all other algorithms with any value of k; this means the K-Means DECT algorithm performs effectively on a large network.*

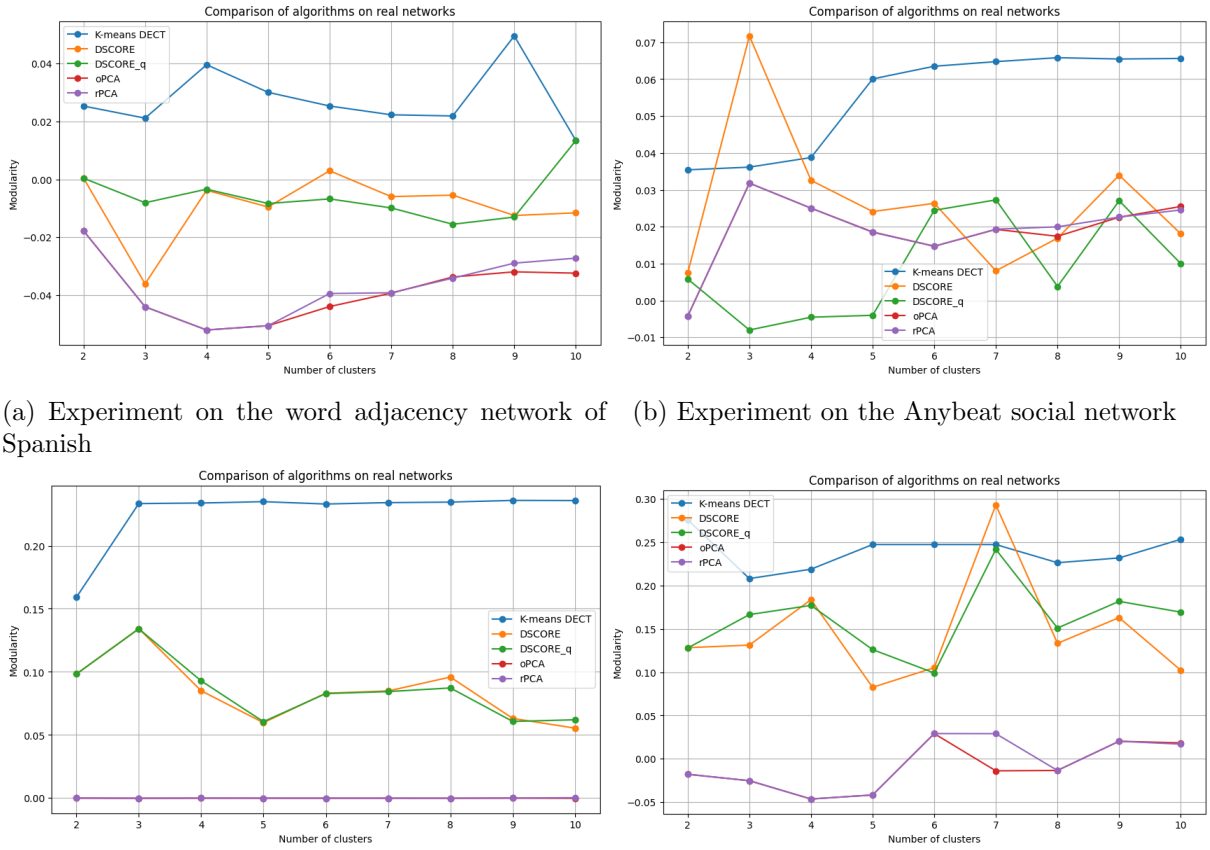
Experiments on Extra-Large Networks

Finally, we test our approach on extra-large networks, where nodes range from 10000 to 26000 and edges from 40000 to 220000.

- *Word adjacency network of Spanish* [16]: This network, denoted as G_{13} ($G_{13} = (11586, 45129)$), represents directed word adjacency in Spanish texts. Nodes are words, and a directed edge from i to j indicates that word j directly follows word i in the text.
- *Anybeat social network (2013)* [9]: This network, denoted as G_{14} ($G_{14} = (12645, 67053)$), is a snapshot of the Anybeat online social network from 2013, captured before the platform was shut down. Nodes represent users, and a directed edge (i, j) indicates that user i follows user j.

- *FOLDOC entries (2002)* [5]: This network, denoted as G_{15} ($G_{15} = (13356, 120238)$), maps hyperlinks among entries in the Free Online Dictionary of Computing (FOLDOC). A directed edge (i, j) indicates that term j is referenced in the entry for term i. Edge weights represent the number of references.

- *Google internal webpages (2007)* [17]: This network, denoted as G_{16} ($G_{16} = (15763, 171206)$), represents the hyperlink structure among Google's internal webpages. A directed edge (i, j) indicates that webpage i contains a hyperlink to webpage j.



(a) Experiment on the word adjacency network of Spanish

(b) Experiment on the Anybeat social network

(c) Experiment on the FOLDOC entries network

(d) Experiment on the Google internal webpages network

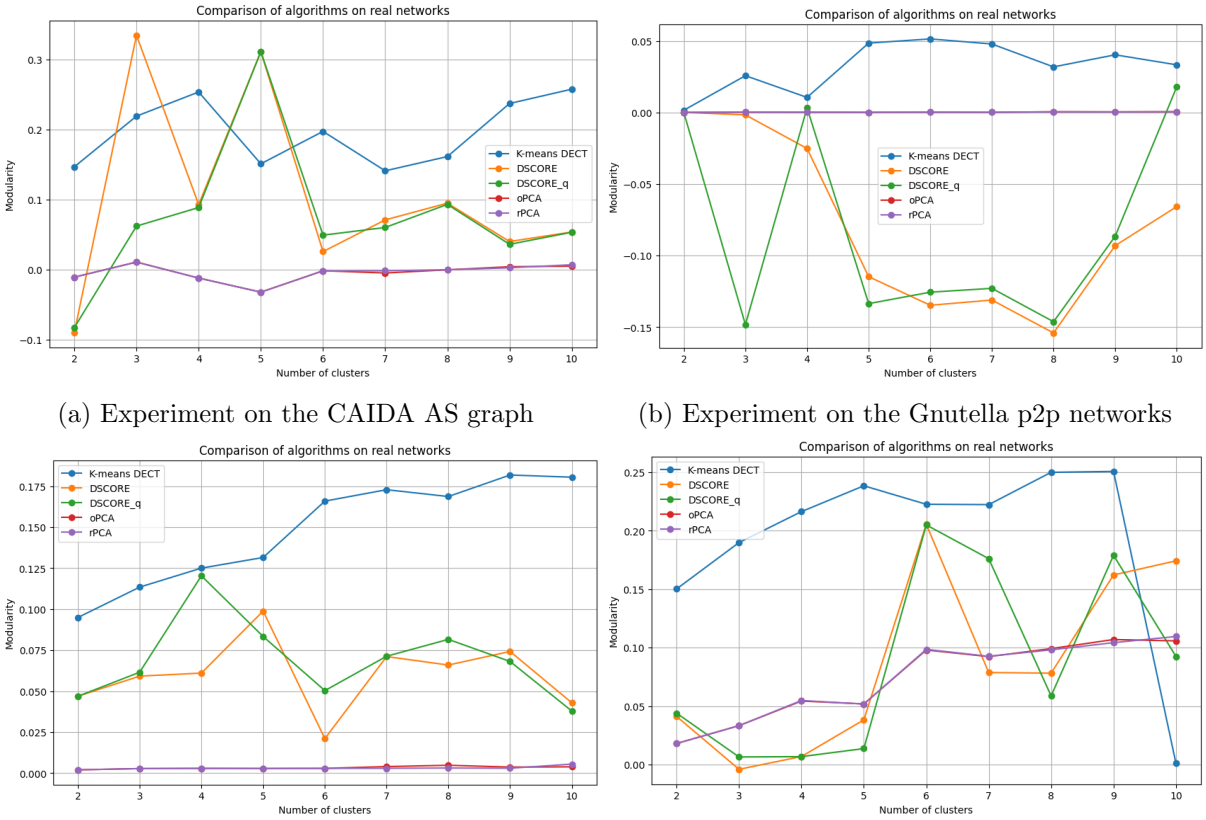
Figure 4: Results of the K-Means DECT algorithm on real-world networks

- *CAIDA AS graph (2005)* [1]: This network, denoted as G_{17} ($G_{17} = (20037, 80948)$), represents Autonomous System (AS) relationships on the Internet in 2005. The relationships were inferred using the Serial-1 method from RouteViews BGP table snapshots combined with a set of heuristics.

- *Gnutella p2p network (2002)* [19]: This network, denoted as G_{18} ($G_{18} = (22687, 54705)$), is a snapshot of the Gnutella peer-to-peer file sharing network taken on August 25, 2002. Nodes represent hosts in the network topology, and directed edges denote connections between them.

- *CORA citations (1998)* [15]: This network, denoted as G_{19} ($G_{19} = (23166, 91500)$), captures citation relationships among papers indexed by CORA, an early computer science research paper search engine. A directed edge (i, j) indicates that paper i cites paper j within this dataset. Papers not in the dataset are excluded, and self-loops may be present.

- *SCOTUS majority opinions* [11]: This network, denoted as G_{20} ($G_{20} = (25417, 216738)$), represents legal citations among majority opinions authored by the Supreme Court of the United States (SCOTUS) between 1754 and 2002 (2008 version). Node metadata includes detailed descriptions of each opinion in addition to the citation network.



(a) Experiment on the CAIDA AS graph

(b) Experiment on the Gnutella p2p networks

(c) Experiment on the CORA citations network

(d) Experiment on the SCOTUS majority opinions network

Figure 5: Results of the K-Means DECT algorithm on real-world networks

Remark 4.4 *When working with extremely large graphs, the K-Means DECT algorithm remains superior to most algorithms for nearly all values of k, except for a few cases where its performance is slightly less effective, though the difference is insignificant. We observed that the K-Means DECT algorithm demonstrates solid and stable clustering performance on networks with clear community structures, outperforming the other four algorithms on such networks in almost a value of k, except for some cases in which the community structure is unclear.*

4.3 Conclusion of the experiments

In general, the value of modularity correspondence to the clustering result of K-Means DETC is higher than another algorithm like oPCA, rPCA, DScore, or DScoreq in three kinds of real networks (small networks, extensive networks, and extra large networks). Except for some exceptional cases (with the value of k such that the network's community structure with that value k is faint and unclear), the modularity corresponds to the algorithm K-Means DETC performs worse than other algorithms.

5 Conclusion

In this paper, we introduced the Directed Euclidean Commute Time (DECT) as an extension of the Euclidean Commute Time (ECT) for directed graphs, providing a robust metric to measure distances in such networks. By integrating DECT with the K-Means clustering algorithm, we developed a novel approach that effectively captures directed graphs' asymmetry and relational structure. Furthermore, we enhanced the computational efficiency of DECT by leveraging its connection to the Singular Value Decomposition (SVD) of the Diplacian matrix.

Experiments on real-world networks demonstrated the effectiveness of the proposed method in detecting meaningful communities, consistently achieving higher accuracy than conventional distance-based clustering techniques. These results underscore the value of leveraging hitting times as a distance measure for analyzing complex networks.

Acknowledgments

This research was supported by the International Centre of Research and Postgraduate Training in Mathematics, Institute of Mathematics, project code: ICRTM03-2024.01

References

[1] The caida as relationships dataset. Available at `http://www.caida.org/data/as-relationships/`.

[2] Lada A. Adamic and Natalie S. Glance. The political blogosphere and the 2004 u.s. election: divided they blog. In *LinkKDD '05*, 2005.

[3] United States Federal Aviation Administration. "air traffic control system command center.". Available at `http://www.fly.faa.gov/`.

[4] Alex Arenas, Jordi Duch, Alberto Fernández, and Sergio Gómez. Size reduction of complex networks preserving modularity. *New Journal of Physics*, 9:176 – 176, 2007.

[5] Vladimir Batagelj, Andrej Mrvar, and Matjaz Zaversnik. Network analysis of texts. 2002.

[6] Emmanuel J. Candès and Benjamin Recht. Exact matrix completion via convex optimization. *Foundations of Computational Mathematics*, 9:717–772, 2008.

[7] Tien Dat Dang, Duy Hieu Do, and Thi Ha Duong Phan. Community detection in directed graphs using stationary distribution and hitting times methods. *Social Network Analysis and Mining*, 13:1–30, 2023.

[8] Manlio De Domenico, Mason A. Porter, and Alex Arenas. Multilayer analysis and visualization of networks. *ArXiv*, abs/1405.0843, 2014.

[9] Michael Fire, Rami Puzis, and Yuval Elovici. Link prediction in highly fractional data sets. 2013.

[10] Santo Fortunato. Community detection in graphs. *Physics reports*, 486(3-5):75–174, 2010.

[11] James H. Fowler and Sangick Jeon. The authority of supreme court precedent. *Soc. Networks*, 30:16–30, 2008.

[12] Stephen Kosack, Michele Coscia, Evann Smith, Kim Albrecht, Albert Ĺaszló Barabási, and Ricardo Hausmann. Functional structures of us state governments. *Proceedings of the National Academy of Sciences of the United States of America*, 115:11748 – 11753, 2018.

[13] Yanhua Li and Zhi-Li Zhang. Digraph laplacian and the degree of asymmetry. *Internet Mathematics*, 8:381 – 401, 2012.

[14] Paolo Massa, Martino Salvetti, and Danilo Tomasoni. Bowling alone and trust decline in social network sites. *2009 Eighth IEEE International Conference on Dependable, Autonomic and Secure Computing*, pages 658–663, 2009.

[15] Andrew McCallum, Kamal Nigam, Jason D. M. Rennie, and Kristie Seymore. Automating the construction of internet portals with machine learning. *Information Retrieval*, 3:127–163, 2000.

[16] Ron Milo, Shalev Itzkovitz, Nadav Kashtan, Reuven Levitt, Shai S. Shen-Orr, Inbal Ayzenshtat, Michal Sheffer, and Uri Alon. Superfamilies of evolved and designed networks. *Science*, 303:1538 – 1542, 2004.

[17] Gergely Palla, Illés J. Farkas, Péter Pollner, Imre Derényi, and Tam'as Vicsek. Directed network modules. *New Journal of Physics*, 9:186 – 186, 2007.

[18] Pascal Pons and Matthieu Latapy. Computing communities in large networks using random walks. In *J. Graph Algorithms Appl.*, 2004.

[19] Matei Ripeanu, Ian T Foster, and Adriana Iamnitchi. Mapping the gnutella network: Properties of large-scale peer-to-peer systems and implications for system design. *ArXiv*, cs.DC/0209028, 2002.

[20] Erik Gabriel Schwimmer. Exchange in the social structure of the orokaiva: Traditional and emergent ideologies in the northern district of papua. 1970.

[21] Shai S. Shen-Orr, Ron Milo, Shmoolik Mangan, and Uri Alon. Network motifs in the transcriptional regulation network of escherichia coli. *Nature Genetics*, 31:64–68, 2002.

[22] Christiane Takacs. On the fundamental matrix of finite state markov chains, its eigensystem and its relation to hitting times. *Mathematica Pannonica*, 17:183–193, 2006.

[23] John White, Erica L. Southgate, J. Nichol Thomson, and Sydney Brenner. The structure of the nervous system of the nematode caenorhabditis elegans. *Philosophical transactions of the Royal Society of London. Series B, Biological sciences*, 314 1165:1–340, 1986.

[24] Luh Yen, Denis Vanvyve, Fabien Wouters, François Fouss, Michel Verleysen, and Marco Saerens. Clustering using a random walk based distance measure. In *The European Symposium on Artificial Neural Networks*, 2005.

[25] Lovro Šubelj and Marko Bajec. Software systems through complex networks science: review, analysis and applications. *Proceedings of the First International Workshop on Software Mining*, 2012.