UCL Université catholique de Louvain

École polytechnique de Louvain (EPL)

epl

Distance measures for graph theory :

Comparisons and analyzes of different methods

Dissertation presented by
Maxime DUYCK

for obtaining the Master's degree in
Mathematical Engineering

Supervisor(s)
Marco SAERENS

Reader(s)
Guillaume GUEX, Bertrand LEBICHOT

Academic year 2016-2017

Acknowledgments

First, I would like to thank my supervisor Pr. Marco Saerens for his presence, his advice and his precious help throughout the realization of this thesis.

Second, I would also like to thank Bertrand Lebichot and Guillaume Guex for agreeing to read this work.

Next, I would like to thank my parents, all my family and my friends to have accompanied and encouraged me during all my studies.

Finally, I would thank Malian De Ron for creating this template [65] and making it available to me. This helped me a lot during *"le jour et la nuit"*.

Contents

Introduction 1

Contents

1.1. Context presentation

The analysis of networks or graphs is a highly researched field in the areas of applied mathematics and computing. Recent development in IT resources considerably increased calculating power and thus the ability to tackle problems that can be represented as enormous networks. This leads to the extended usage of graphs in various areas of science, such as biology, chemistry, physics, pattern recognition, data mining and machine learning [6, 28, 51, 59, 73, 99].

Broadly speaking, a graph is a mathematical structure that contains a certain number of elements called nodes. These can be paired using links known as edges if a relationship exists between them. For instance, the most popular network used by millions of people daily, the World Wide Web, has pages connected by hyperlinks. Another well-known example is the map of telephone communications, which is often used to introduce the concept of clustering. The aim of clustering is to gather the nodes into groups that represent real-world communities. To do this, it may utilize all information contained in the graphical framework.

One of the problems with the process of clustering is how to quantify the nodal connections in a network to construct communities. This can be solved by exploring the distance between vertices. Indeed, it is precisely the main subject of this thesis, and we attempt to address this issue using the best possible approach.

There are several methods to define nodal distance, some of which are presented below. Originally, the scientific research focused mainly on two basic techniques, the shortest-path and the resistance distances. The latter is also called the commute-time distance. However, they suffer from some significant flaws [62, 63]. The former depends only on the most direct routes and thus fails to integrate the "degree of connectivity" between two nodes. This means that if the shortest-path distance between two pairs of vertices is constant, then two nodes connected by one path should be considered as similar as any other pair joined by several routes.

Another limitation emerges in computing the interval from a given node, the shortest-path method usually provides many ties, or equidistant vertices [32]. In short, this approach fails to take the entire graphical structure into account [62, 63]. On the other hand, the commute-time distances converge to yield a useless value, depending only on the degrees of the two vertices when the size of the network increases : *the random walker becomes "lost in space" because the Markov chain mixes too quickly* [62, 63].

Recent papers have introduced various distance measures that try to avoid these shortcoming. This thesis presents some of these methods before attempting to identify the most ideal techniques. The two main contributions can be therefore summarized as :

- *First, an exhaustive experimental comparison between the different similarity or dissimilarity measures presented in [29].*
- *Next, a experimental comparison between the best methods of [29] and a brand new method, the randomized optimal transport introduced in [38].*

To this effect, I first present the contents of my study in the next section.

1.2. Contents

Following the introduction, Chapter 2 addresses the theoretical aspect of the topic. First, I reitarate some basic and useful concepts about graphs to etablish an agreement on the relevant definitions. Afterwards, I describe most of the distance measures presented in [29]. Finally, I present a detailed discussion about an original solution introduced by [38].

Next, in Chapter 3, I outline the procedures used to test the different methods. This section also explains the statistical tools for comparing them and presents the datasets on the graphs utilized for the evaluation.

Chapter 4 focuses on the results and attempts to make inferences using the tools introduced in the previous segment.

Finally, I explain the limitations of my work and possibilities for further investigations in Chapter 5 before concluding in Chapter 6.

Theoretical part

2

Contents

2.1. Preliminaries

This chapter begins with a short theoretical prelude about the fundamentals of networks or graphs. I then describe the matrices and tools used in the remainder of this thesis. This section is largely inspired by [9, 29, 35, 51, 73, 86] and readers are encouraged to consult original sources for further details.

2.1.1. Networks and graphs

A network or graph $\mathrm{G} = (\mathcal{V}, \mathcal{E})$ is a mathematical structure that contains

- A finite, nonempty set $\mathcal{V}$ of n elements known as nodes (or vertices); it represents objects or entities in the real world and,
- a set $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ of m elements called edges (or links) which couple nodes. Links represent the existence of a connection between two objects.

As said before, there are various examples of graphs. One is a social network, where the entities are the users and the edges are the relationships between them. Another case can be that of molecules (where the atoms and the chemical bonds are the vertices and the edges, respectively) or subway maps (with the stations and the lines).

First, a network is *undirected* if the edges have no orientation; this means that for all links $(i, j) \in \mathcal{E}$, the edges (i, j) and (j, i) are identical. Conversely, $(i, j) \neq (j, i)$ for a *directed* plot. Second, in a *weighted* graph, the weight w_{ij} is associated with each link represents the affinity between the node i and the node j. Third, a network is *simple* if there is at most a single edge between two vertices and if it has no self-loop (edge starting and ending at the same node). In the following sections, I only consider the last case.

2.1.2. Useful matrices and tools

Adjacency matrix

The adjacency matrix of a graph G is an $n \times n$ matrix containing its structure. For an unweighted network, the entries a_{ij} take binary values and are defined as

$$a_{ij} = [\mathrm{A}]_{ij} \triangleq \begin{cases} 1 & \text{if } i \to j \in \mathcal{E} \\ 0 & \text{otherwise.} \end{cases} \tag{2.1}$$

For its weighted counterpart, we can identify two different adjacency matrices. The former is defined as before and a_{ij} records the presence or the absence of a link between i and j. The latter considers the affinities w_{ij} between nodes and is described by

$$a_{ij} = [\mathrm{A}]_{ij} \triangleq \begin{cases} w_{ij} & \text{if } i \to j \in \mathcal{E} \\ 0 & \text{otherwise.} \end{cases} \tag{2.2}$$

Note that the adjacency matrix of an undirected graph is always symmetric. As there are simple networks, the diagonal of A is filled with zeroes.

Volume of a graph

The volume of G is simply the sum of all its affinities. To obtain this value, it is sufficient to add all the entries in the adjacency matrix:

$$\mathrm{vol(G)} = \sum_{i,j=1}^{n} a_{ij} = a_{\bullet\bullet}. \tag{2.3}$$

Diagonal degree matrix

From the adjacency matrix, it is also possible to compute the degree of each node. For an undirected graph G, the sum of row i, $\sum_{j=1}^{n} a_{ij} = a_{i\bullet}$ is equal to that of column i, $\sum_{j=1}^{n} a_{ji} = a_{\bullet i}$. These totals correspond to d_i, the degree of vertex i. In matrix form, we can derive the $n \times 1$ degree vector from

$$d \triangleq \mathrm{A}e, \tag{2.4}$$

where e is a unit column vector of length n (i.e., $e = [1, 1, \ldots, 1]^{\mathrm{T}}$).

In the directed case, the sums of the ith row and of the ith column are no longer equal. We refer to the latter value as the indegree of node i and the former as its outdegree. In matrix form, these two vectors are respectively written as

$$d_i \triangleq \mathrm{A}^{\mathrm{T}} e, \tag{2.5}$$

$$d_o \triangleq \mathrm{A}e. \tag{2.6}$$

Finally, the diagonal degree matrices are determined by placing the elements of the degree vectors on their diagonals. For directed graphs, we obtain $\mathrm{D}_i = \mathrm{Diag}(d_i)$ and $\mathrm{D}_o = \mathrm{Diag}(d_o)$; for the undirected scenario, it is simply $\mathrm{D} = \mathrm{Diag}(d)$.

Cost matrix

In addition to the affinities within the adjacency matrix, non-negative costs are sometimes assigned to the edges of G. This can be regarded as the difficulty in moving from one side of the link to the other. In some cases, we may compute the costs from the affinities by setting $c_{ij} = 1/a_{ij}$. We defined this matrix as

$$[\mathrm{C}]_{ij} \triangleq \begin{cases} c_{ij} & \text{if } i \to j \in \mathcal{E} \\ \infty & \text{otherwise.} \end{cases} \tag{2.7}$$

Transition matrix

Assuming that each node has at least one outgoing link, we can allocate the transition probabilities, which are calculated [37, 49, 74] by

$$p_{ij} \triangleq \frac{a_{ij}}{\sum_{j'=1}^{n} a_{ij'}} = \frac{a_{ij}}{a_{i\bullet}}. \tag{2.8}$$

These quantities may be interpreted as the likelihood that a random walker on a graph at node i travels towards j. We can add the probabilities that are directly proportional to the affinities of edges $i \to j$. In matrix form, this generate

$$\mathrm{P} = \mathrm{D}_0^{-1}\mathrm{A}, \tag{2.9}$$

where P is the stochastic **transition matrix**.

Laplacian matrix and its normalized variant

The Laplacian matrix is a quantity appearing in two configurations across various concepts. The elements of the unnormalized variety for simple undirected graphs are characterized by

$$l_{ij} \triangleq \begin{cases} d_i & \text{if } i = j \\ -a_{ij} & \text{otherwise.} \end{cases} \tag{2.10}$$

In matrix form, this becomes

$$\mathrm{L} \triangleq \mathrm{D} - \mathrm{A}. \tag{2.11}$$

Some properties of the above matrix are important. First, if the graph is symmetric, then so is L, which is ranked $n-1$ [18]. Next, this matrix is doubly centered; if $\mathbf{0}$ is a column vector made of zeroes then $\mathrm{L}e = \mathbf{0}$ and $e^{\mathrm{T}}\mathrm{L} = \mathbf{0}^{\mathrm{T}}$ (the sum of the elements in each row and column of L is equal to zero).

The normalized $\tilde{\mathrm{L}}$ is defined [1, 18, 98] as

$$\tilde{\mathrm{L}} \triangleq \mathrm{D}^{-\frac{1}{2}}\mathrm{L}\mathrm{D}^{-\frac{1}{2}} = \mathrm{I} - \mathrm{D}^{-\frac{1}{2}}\mathrm{A}\mathrm{D}^{-\frac{1}{2}}, \tag{2.12}$$

or elementwise as

$$\tilde{l}_{ij} \triangleq \begin{cases} 1 - \frac{a_{ij}}{\sqrt{d_i d_j}} & \text{if } i = j \\ -\frac{a_{ij}}{\sqrt{d_i d_j}} & \text{otherwise.} \end{cases} \tag{2.13}$$

Pseudoinverse of the Laplacian matrix

The final matrix that plays an important role in the following chapters is the *Moore-Penrose pseudoinverse of the Laplacian matrix*. The pseudoinverse is the generalization of the inverse matrix in situations where the latter is not well defined - i.e., for rank-deficient cases

and non-square matrices [4, 5, 7, 8, 42, 80, 85]. As previously mentioned, the Laplacian matrix has rank $n-1$ and thus we have to discuss the pseudoinverse. There are in fact several types of thereof and Moore-Penrose is the one of interest; it is based on singular value decomposition (SVD).

Since the Laplacian matrix is real, symmetric and positive semi-definite, its spectral decomposition is also an SVD [5]. Thus, noting the eigenvalues and eigenvectors of L as $\{\lambda_k, u_k\}_{k=1}^{n}$, its pseudoinverse is expressed as

$$\mathrm{L}^{+} = \sum_{k=1}^{n-1} \frac{1}{\lambda_k} u_k u_k^{\mathrm{T}}. \tag{2.14}$$

As it is also doubly centered, the Moore-Penrose pseudoinverse can be computed by the formula [80]

$$\mathrm{L}^{+} = \left(\mathrm{L} + \frac{ee^{\mathrm{T}}}{n}\right)^{-1} - \frac{ee^{\mathrm{T}}}{n}. \tag{2.15}$$

2.2. Distances and kernels on a graph

This section briefly recapitules the notions of (dis)similarity measures, distances and kernels on a graph. It is also largely inspired by [29], in which the interested reader can obtain more details.

2.2.1. Notion of (dis)similarity measures

Similarity measure is the ability to determine if two objects are alike [24, 83, 93]. The simple, underlying intuitions are summarized in [61] whose definitions are adopted :

- The similarity between two entities is related to their commonality; the more commonality they share, the more similar they are.
- Symmetrically, the reverse is true. The similarity between two entities is related to the difference between them; the higher the difference between two objects, the less similar they are.
- Maximum similarity is achieved when the items are identical, no matter how much they have in common.

Mathematically, the notion of dissimilarity is more standardized than that of similarity (see, for example, [24, 36, 43, 57, 93]). Let Δ be a dissimilarity measure over a set of objects, and let Δ_{ik} be that between the objects i and k; Δ_{ik} has to be :

- Nonnegative : $\Delta_{ik} \geq 0$ for all i, k.
- Symmetric : $\Delta_{ik} = \Delta_{ki}$ for all i, k.
- Calibrated to zero : $\Delta_{ik} = 0$ if and only if the objects i and k are identical.

The last condition implies that two different entities have a strictly positive dissimilarity, and that $\Delta_{ii} = 0$ for all i (reflexivity). It may be added that if, in addition, the measure satisfies the triangle inequality

- $\Delta_{ik} \leq \Delta_{ij} + \Delta_{jk}$ for all i, j, k,

then it is known as a **distance measure** [24, 36, 43, 57, 93].

On the other hand, the similarity level is less standardized. Let s be a similarity measure over a set of objects, and let s_{ik} be that between i and k. The first condition is denoted by [36]

- s indicates a symmetric relationship : $s_{ik} = s_{ki}$ for all i.

In addition, [68] includes three reasonable properties:

- $s_{ik} > 0$ for all i, k.
- s_{ik} increases with the similarity between i and k.
- $s_{ik} \leq s_{kk}$ for all i, k.

In fact, some families of similarity may violate one of the above. For instance, the Jaccard index [41] is bounded by unity, $0 \leq s_{ik} \leq 1$, and the cosine similarity [27] satisfies $-1 \leq s_{ik} \leq 1$.

2.2.2. Kernel on a graph

Kernels on graphs are useful mathematical tools for capturing the similarity between sets of nodes. The present experiments derive kernels from each distance measure to obtain the best similarity measurements. This section recalls the basic theory and properties about kernel matrices. For more information, the interested reader may refer to [53, 84, 87].

A kernel is a function that maps two objects to a real number, thus capturing the similarity between them. If the entities and their characteristics are denoted by vectors, their similarity is the results of an inner product between these representations. The vectors do not need to be computed, because the kernel function captures the outcome directly. In our case, the objects are the nodes of a graph, and their similarity is induced by the network structure.

Mathematically, we have a function:

$$k(i, j) : \mathcal{V} \times \mathcal{V} \to \mathbb{R}, \tag{2.16}$$

that returns a real number R for two objects $i, j \in \mathcal{V}$. After calculating R for each pair $(i, j) \in \mathcal{V} \times \mathcal{V}$, the results are placed in a grid $\{[\mathrm{K}]_{ij} = k_{ij} = k(i, j)\}_{i,j=1}^{n}$ called the kernel matrix, which must satisfy two conditions [84, 87].

- K must be symmetric : $k_{ij} = k_{ji}$ for all $i, j \in \mathcal{V}$ and
- K must be positive semi-definite : $x^{\mathrm{T}}\mathrm{K}x \geq 0$ for all $x \in \mathbb{R}^n$.

2.2.3. The shortest-path distance

To illustrate the concept of distance between two nodes on a graph, we briefly recall the most popular measure: the shortest-path or geodesic distance. As mentioned before; however, this method suffers from many disadvantages, since it does not consider the entire structure of the network. For this reason, the approach is not tested in our experiments.

The shortest path distance between each pair of vertices is simply the set of edges connecting them at minimum cost. Those with no paths joining them are disconnected, and the smallest distance between them is infinity. When the values are computed, they are recorded in a distance matrix Δ_{SP}.

2.3. Kernels from distances

Our investigation requires the centered kernel matrices, which is determined directly by some of the (dis)similarity measures. On other cases, a kernel matrix can be obtained from a Euclidean distance measure. This can be done in two ways, both of which are evaluated in the study; they are described below.

2.3.1. Multidimensional scaling

The first method is based on multidimensional scaling [10, 20]. We can obtain a centered kernel matrix K from a distance Δ as follows:

$$\mathrm{K}^{\mathrm{mds}} = -\frac{1}{2}\mathrm{H}\Delta^{(2)}\mathrm{H} \tag{2.17}$$

where $\mathrm{H} = (\mathrm{I} - \frac{ee^{\mathrm{T}}}{n})$ is the centering matrix and $\Delta^{(2)}$ is the elementwise squared distance.

2.3.2. Gaussian mapping

Another way to convert a distance into a kernel is Gaussian mapping [84], which is defined by

$$\mathrm{K}^{g} = \exp\left[-\Delta^{(2)}/2\sigma^{2}\right]. \tag{2.18}$$

The last step to do after these transformations is to ensure that the kernel matrices are positive semi-definite. For this, it is possible to remove the negative eigenvalues from the spectral decomposition [68]; this is applied in all the experiments in this thesis.

2.4. Similarity measures between nodes

In this section, I briefly outline some similarity measures between the nodes of a graph; these are located in Chapter 2 of [29]. Only the methods for which [29] provides an algorithm are

described here and tested in my investigation. Again, more explanations, properties and examples are given in [29]. The MATLAB™ code for implementing the procedures is in the appendices.

2.4.1. Katz index and its Leicht's extension

this technique, proposed in [48], accounts for the number of indirect links between vertices, in addition to the direct ones. The Katz similarity matrix is defined by

$$K_{Katz} \triangleq \alpha A + \alpha^2 A^2 + \ldots + \alpha^\tau A^\tau + \ldots = \sum_{t=1}^{\infty} \alpha^t A^t = (I - \alpha A)^{-1} - I \tag{2.19}$$

where A is the adjacency matrix, and α is a discounting factor that has the influence of the "likelihood of effectiveness of a single link" [48]. The parameter may be viewed as the attenuation in the link; $\alpha = 0$ corresponds to a complete attenuation while $\alpha = 1$ is the absence thereof.

An extension subsequently proposed by Leicht and al. in [58] consists of the Katz index divided by the degrees of the starting and ending nodes; this is represented as

$$K_{Leicht} \triangleq D^{-1}(I - \alpha A)^{-1} D^{-1}. \tag{2.20}$$

Algorithm 2.2 of [29] describes the two measures above.

2.4.2. Commute-time distance and Euclidean commute-time distance

The commute-time distance $n(i, j)$ between the nodes i and j is defined as the average number of steps that a random walker, starting at $i \neq j$, needs to take to reach j for the first time before returning to its starting node i :

$$n(i, j) = m(i, j) + m(j, i).$$

In the above formula $m(i, j)$ is the average first-time passage [11, 49, 74, 92]. Both of these quantities are global dissimilarity measures [60] that only integrate the degree of connectedness. In matrix form, the commute-time distance can be written as

$$\Delta_{CT} = \mathrm{vol}(G)\left(\mathrm{diag}(L^+)e^T + e(\mathrm{diag}(L^+))^T - 2L^+\right). \tag{2.21}$$

Moreover, because of the positive semi-definiteness of L^+, the elementwise square root of this interval is also a measure of distance that has as particularity to be Euclidean. The Euclidean commute-time distance is thus denoted by

$$\Delta_{ECT} = \Delta_{CT}^{(\frac{1}{2})} = \left[\mathrm{vol}(G)\left(\mathrm{diag}(L^+)e^T + e(\mathrm{diag}(L^+))^T - 2L^+\right)\right]^{(\frac{1}{2})}. \tag{2.22}$$

More recently, a corrected version of these two measures was proposed in [62]. The underlying principle is to express the initial measure as a series of terms of decreasing significance

before removing the two, which results in an inconvenience. In this manner, we obtain

$$\begin{aligned} \Delta_{\mathrm{CCT}} &= \Delta_{\mathrm{CT}} - \mathrm{vol}(\mathrm{G})\left[\mathrm{diag}(\mathrm{D}^{-1}+\mathrm{D}^{-1}\mathrm{A}\mathrm{D}^{-1})e^{\mathrm{T}}\right. \\ &\quad \left. + e(\mathrm{diag}(\mathrm{D}^{-1}+\mathrm{D}^{-1}\mathrm{A}\mathrm{D}^{-1}))^{\mathrm{T}} - 2(\mathrm{D}^{-1}+\mathrm{D}^{-1}\mathrm{A}\mathrm{D}^{-1})\right] \qquad (2.23) \\ \Delta_{\mathrm{CECT}} &= \Delta_{\mathrm{CCT}}^{(\frac{1}{2}).} \qquad (2.24) \end{aligned}$$

Algorithm 2.3 of [29] computes these four distances.

2.4.3. SimRank similarity measure

SimRank [45] generalizes the co-citation matrix[1] [88]. It can be calculated by induction with the initialization $\mathrm{K}(0) = \mathrm{I}$ as follows:

$$\begin{cases} \mathrm{K}'(t+1) &= \alpha \mathrm{Q}^{\mathrm{T}} \mathrm{K}(t) \mathrm{Q}, \\ \mathrm{K}(t+1) &= \mathrm{K}'(t+1) - \mathrm{Diag}(\mathrm{K}'(t+1)) + \mathrm{Diag}(d_i > 0). \end{cases} \qquad (2.25)$$

The above is iterated until convergence occurs. The matrix Q contains the elements $q_{ii'} = a_{ii'}/a_{\bullet i'}$ when $a_{\bullet i'} \neq 0$ or $q_{ii'} = 0$ otherwise; it can be written as $\mathrm{Q} = \mathrm{A}(\mathrm{Diag}(e^{\mathrm{T}}\mathrm{A}))^{+}$. The results is expressed as **Algorithm 2.4** from [29].

2.4.4. Exponential diffusion kernel and Laplacian exponential diffusion kernel

The following similarity measures are based on kernels on a graph. They have the property that they increase with the number of paths connecting two nodes increases while their lengths decrease. First, we define the exponential diffusion kernel introduced by Kondor and Lafferty in [52] as

$$\mathrm{K}_{\mathrm{ED}} \triangleq \sum_{t=0}^{\infty} \frac{\alpha^t \mathrm{A}^t}{t!} = \mathrm{expm}(\alpha \mathrm{A}). \qquad (2.26)$$

As noted, the element (i, j) of the matrix A^t corresponds to the number of paths of length t between the node i and node j. Thus, the kernel considers all paths between the nodes, discounting them according to t with a factor of $\alpha^t/t!$.

The Laplacian exponential diffusion kernel introduced in [52, 89] is an alternative to K_{ED} that substitutes the opposite of the Laplacian matrix for the adjacency one. It is expressed as

$$\mathrm{K}_{\mathrm{LED}} \triangleq \mathrm{expm}(-\alpha \mathrm{L}). \qquad (2.27)$$

Algorithm 2.6 of [29] addresses this case.

[1] aslo known as bibliographic coupling, the co-citation score compares two papers and considers they are similar if the two documents are both cited by a lot of common documents ($\mathrm{sim}_{\text{co-cite}}(i,k) = [\mathrm{A}^{\mathrm{T}}\mathrm{A}]_{ik}$).

2.4.5. Modified regularized Laplacian kernel

To describe this type of kernel, we begin by introducing the original version [13, 14, 40, 89] computed by

$$K_{RL} \triangleq \sum_{t=0}^{\infty} \alpha^t (-L)^t = (I + \alpha L)^{-1}, \tag{2.28}$$

with $0 < \alpha < \rho(L)^{-1}$.

To obtain the modified form, it is necessary to present the parameter γ, which controls the importance and relatedness [40]. This modifies the Laplacian matrix such that $L_\gamma \triangleq \gamma D - A$, where $0 < \gamma < 1$. Therefore, the measure is defined by

$$K_{MRL} \triangleq \sum_{t=0}^{\infty} \alpha^t (-L_\gamma)^t = (I + \alpha L_\gamma)^{-1} \tag{2.29}$$

Algorithm 2.7 of [29] calculates the modified regularized Laplacian kernel.

2.4.6. Commute-time kernel

This variable [30, 82] takes its name from the average commute-time $n(i, j)$. It applies the transformation explained in section 2.3.1 to equation 2.21:

$$\begin{aligned} K &= -\frac{vol(G)}{2} H \left(diag(L^+) e^T + e(diag(L^+))^T - 2L^+ \right) H \\ &= -\frac{vol(G)}{2} (-2HL^+H) \\ &= vol(G) L^+, \end{aligned}$$

where $He = 0$ and $HL^+H = L^+$ are used, because the Moore-Penrose pseudoinverse is already centered. The elements of L^+ can measure the similarity between the nodes up to a scaling factor. From there, we simply formulate the commute-time kernel as

$$K_{CT} \triangleq L^+. \tag{2.30}$$

As in the case of distance, a corrected version of this kernel is proposed in[62]. this is written as

$$K_{CCT} = HD^{-\frac{1}{2}} (I - M)^{-1} M D^{-\frac{1}{2}} H, \tag{2.31}$$

with $M = D^{-\frac{1}{2}} \left(A - \frac{dd^T}{vol(G)} \right) D^{-\frac{1}{2}}$.

Algorithm 2.8 of [29] represents this situation.

2.4.7. Regularized commute-time kernel and random walk with restart similarity

The next two similarity measures are based on a discrete diffusion process. The first, the regularized commute-time kernel from [102], is linked with the commute-time kernel. Instead of taking the pseudoinverse of the Laplacian matrix, however, we only apply a simple regularization, which leads to

$$\mathrm{K}_{\mathrm{RCT}} \triangleq (\mathrm{D} - \alpha \mathrm{A})^{-1}, \tag{2.32}$$

with $\alpha \in]0,1[$.

The random walk with restart similarity [76, 94, 95] is inspired by the PageRank algorithm used by Google [12, 54, 75]. Where it differs from the previous case is at each step, the random walker has the possibility of restarting at the initial node i. The final situation is depicted as follows:

$$\mathrm{K}_{\mathrm{RWR}} \triangleq (\mathrm{I} - \alpha \mathrm{P})^{-1} = (\mathrm{D} - \alpha \mathrm{A})^{-1}\mathrm{D}. \tag{2.33}$$

We should note that this is not a valid kernel and has to be transformed into one for our investigation. It is illustrated by **Algorithm 2.9** from [29].

2.4.8. Markov diffusion distance and kernel

The main idea here is to develop a valid kernel by adapting the definition of diffusion distance [19, 70, 71, 78, 79] to discrete-time processes and periodic Markov chains. The Markov diffusion distance was originally proposed almost simultaneously by Coifman, Nadler et al. in [19, 70] and Pons and Lapaty [78] as

$$\Delta_{ij}^2(t) \triangleq \sum_{k=1}^{n} w_k (x_{ik}(t) - x_{jk}(t))^2 \text{ with } w_k = \frac{1}{\pi_k}. \tag{2.34}$$

The variable $x_{ik}(t) = \mathrm{P}(s(t) = k | s(0) = i)$ corresponds to the probability that the random walker starting from i is at k after t steps. Moreover, $w_k > 0$ is a weighting factor associated at each state k equal to the inverse of the stationary distribution of the Markov chain, π.

This definition, however, is not appropriate for periodic Markov chains, and it is therefore preferable to take the mean quantity over a time window t. The result is the definition below for the original and time-averaged Markov diffusion square distances:

$$\begin{aligned} \Delta_{\mathrm{MD}}^{(2)}(t) &= \mathrm{diag}\left(\mathrm{Z}(t)\mathrm{D}_w\mathrm{Z}^{\mathrm{T}}(t)\right) e^{\mathrm{T}} + e\left(\mathrm{diag}\left(\mathrm{Z}(t)\mathrm{D}_w\mathrm{Z}^{\mathrm{T}}(t)\right)\right)^{\mathrm{T}} - 2\mathrm{Z}(t)\mathrm{D}_w\mathrm{Z}^{\mathrm{T}}(t) \\ \text{with } \mathrm{Z}(t) &= \begin{cases} \mathrm{P}^t \text{ for the original definition of the Markov distance} \\ \frac{1}{t}\sum_{\tau=1}^{t}\mathrm{P}^{\tau} \text{ when averaging over a time window,} \end{cases} \end{aligned} \tag{2.35}$$

where $\mathrm{D}_w = \mathrm{Diag}(w)$ is a diagonal matrix containing the weights.

To obtain the Markov diffusion kernel [31], we use adopt the conversion outlined in Section 2.3.1. We obtain

$$\begin{aligned} K_{MD}(t) &= HZ(t)D_w Z^T(t)H \\ \text{with } Z(t) &= \begin{cases} P^t \text{ for the original definition of the Markov distance} \\ \frac{1}{t}\sum_{\tau=1}^{t} P^\tau \text{ when averaging over a time window.} \end{cases} \end{aligned} \tag{2.36}$$

Algorithm 2.10 of [29] computes these measures.

2.5. Families of dissimilarity between nodes

This section explores the families of dissimilarity between nodes from Chapter 3 of [29]. These families have appeared due to the flaws of shortest-path and the commute-time distances noted in the introduction. To fix this problem, several researchers have proposed using distances that interpolate between the two techniques [3, 15, 17, 32, 50, 101]. These families are named for their dependence on a continuous parameter to categorize distances. This parameter controls the interpolation; at one limit of its value, these quantities converge to form the shortest-path distance, while the other end indicates the commute-time distance.

Once again, I have chosen to follow the development of the previous section. Only families of dissimilarity for which [29] provides an algorithm are described here, and the MATLAB™ codes can be found in the appendices. The interested reader is alos encouraged to look for the details in [29] from which this section is largely inspired.

2.5.1. Logarithmic forest distance

The logarithmic forest distance introduced by Chebotarev in [15] is a family of distances whose construction is based on the matrix forest theorem [13, 16]. After defining the regularized Laplacian kernel (see equation 2.28), Chebotarev has computed a new matrix S as follows:

$$\begin{cases} S = (\alpha-1)\log_\alpha K_{RL} & \text{when } \alpha \neq 1 \\ S = \ln K_{RL} & \text{when } \alpha = 1, \end{cases} \tag{2.37}$$

where $\alpha > 0$, and the logarithmic function determines the components of the matrix K_{RL} elementwise. Finally, the logarithmic forest distance matrix is denoted [15] by

$$\Delta_{LF} \triangleq \operatorname{diag}(S)e^T + e\left(\operatorname{diag}(S)\right)^T - 2S, \tag{2.38}$$

where the expression on the right hand side is the standard transformation to obtain a squared distance from a symmetric similarity measure when it is an inner product (inverse transformation from Section 2.3.1). **Algorithm 3.1** from [29] generates the forest distance matrix.

2.5.2. Bag-of-paths probability matrix

All families of distances described henceforth are based on the bag-of-paths framework [32]. This framework is first explained before the families of dissimilarity are defined. The principal idea of the structure is the probability of picking a path starting at node i and ending at j from a bag-of-paths [23, 32, 50, 56, 67] containing objects with the following properties:

- The entities are paths,$\wp$, of arbitrary lengths.
- Paths are sampled from the infinite, countable, bag-of-paths with replacement.
- Each route is weighted according to its total cost, $\tilde{c}(\wp)$. The likelihood of drawing a low-cost path is higher than drawing a high-cost one.

To obtain the probability, we need to use the transition probability matrix P (equations 2.8 and 2.9), known as the reference form $\mathrm{P}^{\mathrm{ref}}$ in this case. We also require the cost matrix C, which contains the costs c_{ij} of each edge and enables computation of the total thereof for a path $\tilde{c}(\wp)$ by simply summing the local costs of the links belonging to the route.

The last step is to find the likelihood distribution on the set of paths. Some calculations reveals a Gibbs-Boltzmann probability distribution [32] defined by

$$\mathrm{P}(\wp) = \frac{\tilde{\mathrm{P}}^{\mathrm{ref}}(\wp)\exp\left[-\theta\tilde{c}(\wp)\right]}{\sum_{\wp'\in\mathcal{P}}\tilde{\mathrm{P}}^{\mathrm{ref}}(\wp')\exp\left[-\theta\tilde{c}(\wp')\right]} = \frac{\tilde{\pi}^{\mathrm{ref}}(\wp)\exp\left[-\theta\tilde{c}(\wp)\right]}{\sum_{\wp'\in\mathcal{P}}\tilde{\pi}^{\mathrm{ref}}(\wp')\exp\left[-\theta\tilde{c}(\wp')\right]}, \tag{2.39}$$

where

$$\tilde{\pi}^{\mathrm{ref}}(\wp_{ij}) \triangleq \prod_{\tau=1}^{t} p^{\mathrm{ref}}_{k_{\tau-1}k_{\tau}} \tag{2.40}$$

is the product of the transition probabilities along path $\wp_{ij}$, whose starting and ending nodes are known, and θ is the parameter that controls the exploration carried out the graph. The greater the value of θ, the smaller the amount of examination performed, and only the lowest-cost routes between vertices are chosen. Conversely, a small θ allows each path to be selected according its likelihood.

We are now able to express the bag-of-paths probability of drawing a path starting from i to j as

$$\mathrm{P}(s=i,e=j) = \frac{\sum_{\wp\in\mathcal{P}_{ij}}\tilde{\pi}^{\mathrm{ref}}(\wp)\exp\left[-\theta\tilde{c}(\wp)\right]}{\sum_{\wp'\in\mathcal{P}}\tilde{\pi}^{\mathrm{ref}}(\wp')\exp\left[-\theta\tilde{c}(\wp')\right]}. \tag{2.41}$$

To compute this in closed form, we need to introduce a new matrix as follows:

$$\mathrm{W} \triangleq \mathrm{P}^{\mathrm{ref}} \circ \exp\left[-\theta\mathrm{C}\right] \tag{2.42}$$

where $\circ$ represents the Hadamard[2] matrix product, and the exponential function is also taken elementwise.

Moreover, if $\theta > 0$ and at least one $c_{ij} > 0$ when $a_{ij} > 0$, the series of powers of W converges and we can pose the fundamental matrix [49]

$$\mathrm{Z} \triangleq \sum_{t=0}^{\infty} \mathrm{W}^t = (\mathrm{I} - \mathrm{W})^{-1}. \tag{2.43}$$

This last matrix allows to compute the bag-of-paths probability: [32] as

$$\mathrm{P}(s = i, e = j) = \frac{z_{ij}}{z_{\bullet\bullet}}, \tag{2.44}$$

with $z_{\bullet\bullet} = \sum_{i,j=1}^{n} z_{ij}$. In matrix form, we have

$$\Pi = \frac{\mathrm{Z}}{z_{\bullet\bullet}}. \tag{2.45}$$

Algorithm 3.2 from [29] determines the bag-of-paths probability matrix.

2.5.3. Bag-of-hitting-paths probability matrix

Here, we restrict the abovementioned distance by only accepting the paths whose terminal node does not appear more than once. The ending vertex has to occur only at the end of the path. It is necessary to modify the fundamental matrix to identify the so-called bag-of-hitting-paths probabilities [32] as follows:

$$\mathrm{Z}_h = \mathrm{Z}\mathrm{D}_h^{-1} \text{ with } \mathrm{D}_h = \mathrm{Diag}(\mathrm{Z}). \tag{2.46}$$

We can now define this probability:

$$\mathrm{P}_h(s = i, e = j) = \frac{\sum\limits_{\wp \in \mathcal{P}_{ij}^h} \tilde{\pi}^{\mathrm{ref}}(\wp) \exp\left[-\theta \tilde{c}(\wp)\right]}{\sum\limits_{i',j'=1}^{n} \sum\limits_{\wp' \in \mathcal{P}_{i'j'}^h} \tilde{\pi}^{\mathrm{ref}}(\wp') \exp\left[-\theta \tilde{c}(\wp')\right]} = \frac{z_{ij}/z_{jj}}{\sum\limits_{i',j'=1}^{n} (z_{i'j'}/z_{j'j'})}. \tag{2.47}$$

In matrix form, this may be written as

$$\Pi_h = \frac{\mathrm{Z}_h}{e^{\mathrm{T}}\mathrm{Z}_h e} = \frac{\mathrm{Z}\mathrm{D}_h^{-1}}{e^{\mathrm{T}}\mathrm{Z}\mathrm{D}_h^{-1} e}. \tag{2.48}$$

This version of the bag-of-hitting-paths probabilities includes the zero-length paths starting and ending at the same node. The other form, which omits these routes, is expressed as

$$\overline{\Pi}_h = \frac{\mathrm{Z}\mathrm{D}_h^{-1} - \mathrm{I}}{e^{\mathrm{T}}(\mathrm{Z}\mathrm{D}_h^{-1} - \mathrm{I})e}. \tag{2.49}$$

Algorithm 3.3 of [29] calculates Π_h and $\overline{\Pi}_h$.

[2] = elementwise

2.5.4. The bag-of-hitting-paths surprisal distance matrix

We can now derive a distance measure from the bag-of-hitting-paths framework introduced in the previous section[3]. This takes the associated weighted surprisal measure, $-\log \mathrm{P}_h(s=i, e=j)$, which quantifies the level of "surprise" generated by the event $(s=i)\bigwedge(e=j)$ as an interval after it has been symmetrized. The surprisal distance can then be defined [32] by

$$\begin{aligned} \Delta_{ij}^h &= \begin{cases} -\dfrac{\log \mathrm{P}_h(s=i,e=j)+\log \mathrm{P}_h(s=j,e=i)}{2} & \text{if } i\neq j \\ 0 & \text{if } i=j \end{cases} \\ &= \begin{cases} -\dfrac{\log\left(\frac{z_{ij}/z_{jj}}{\mathcal{Z}}\right)+\log\left(\frac{z_{ji}/z_{ii}}{\mathcal{Z}}\right)}{2} & \text{if } i\neq j \\ 0 & \text{if } i=j \end{cases}, \end{aligned} \tag{2.50}$$

where $\mathcal{Z}=\sum_{i,j=1}^n \frac{z_{ij}}{z_{jj}}$ is a dissimilarity measure (Section 2.2.1), Δ_{ij}^h is symmetric, $\Delta_{ij}^h \geq 0$ and $\Delta_{ii}^h = 0$ for all i, j. It is also possible to prove that the triangle inequality is satisfied. We therefore conclude that this is an adequate indicator.

Even if the surprisal distance does not depend directly on a parameter, it is nevertheless a family of distance, because it is influenced by the inverse temperature parameter $\theta = 1/\mathrm{T}$, owing to its link to the bag-of-hitting-paths framework.

In matrix form, the surprisal distance can be written as

$$\Delta_h = -\left(\frac{\log \Pi_h + \log \Pi_h^{\mathrm{T}}}{2}\right) - \mathrm{Diag}\left(-\frac{\log \Pi_h + \log \Pi_h^{\mathrm{T}}}{2}\right). \tag{2.51}$$

This is computed using **Algorithm 3.4** of [29].

2.5.5. The bag-of-hitting-paths potential, or free energy, distance matrix

This new distance measure is closely related to the surprisal distance introduced in Section 2.5.4 and to a routing algorithm developed in [91]. It is based on the quantity

$$\phi(i,j) \triangleq -\frac{1}{\theta}\log\frac{z_{ij}}{z_{jj}}, \tag{2.52}$$

which is known as the directed potential distance or the directed free energy distance between i and j [32]. Recall that z_{ij} is an element of the fundamental matrix (equation 2.43).

From this quantity, the potential (or free energy) distance [32, 50] is then

$$\Delta_{ij}^{\phi} \triangleq \begin{cases} \dfrac{\phi(i,j)+\phi(j,i)}{2} & \text{if } i\neq j \\ 0 & \text{if } i=j \end{cases} \tag{2.53}$$

Algorithm 3.5 of [29] can be used to calculate this distance matrix.

[3] Only the version including the zero-length paths.

2.5.6. Randomized shortest-path dissimilarity

Again, based on the bag-of-paths framework, this new dissimilarity interpolates between the shortest-path and half of the commute-time distances [50, 101]. We may say that the walker adopts a "randomized" strategy that is biased towards the routes with lowest cost [81], from which the distinction takes its name.

To compute this measure, we have to determine the randomized shortest-path cost between the two nodes i and j. It is actually the expected value over all hitting paths connecting the vertices, as expressed by

$$\langle \tilde{c} \rangle_{ij} = \mathbb{E}[\tilde{c}] = \sum_{\wp \in \mathcal{P}_{ij}^h} \mathrm{P}(\wp)\tilde{c}(\wp) = \frac{\sum_{\wp \in \mathcal{P}_{ij}^h} \tilde{\pi}^{\mathrm{ref}}(\wp) \exp\left[-\theta \tilde{c}(\wp)\right] \tilde{c}(\wp)}{\sum_{\wp' \in \mathcal{P}_{i'j'}^h} \tilde{\pi}^{\mathrm{ref}}(\wp') \exp\left[-\theta \tilde{c}(\wp')\right]}. \tag{2.54}$$

From there, we define the randomized shortest-path dissimilarity [50, 81, 101] between i and j as the average of the expected cost traveling from i to j and going back again,

$$\Delta_{ij}^{\mathrm{RSP}} \triangleq \frac{\langle \tilde{c} \rangle_{ij} + \langle \tilde{c} \rangle_{ji}}{2}, \text{ for } i \neq j. \tag{2.55}$$

Using algebraic notions, we can rewrite equation 2.54 as

$$\langle \tilde{c} \rangle_{ij} = \frac{e_i^{\mathrm{T}} \mathrm{Z}(\mathrm{C} \circ \mathrm{W})\mathrm{Z} e_j}{z_{ij}} - \frac{e_i^{\mathrm{T}} \mathrm{Z}(\mathrm{C} \circ \mathrm{W})\mathrm{Z} e_j}{z_{ij}}. \tag{2.56}$$

By denoting matrix $\mathrm{S} = (\mathrm{Z}(\mathrm{C} \circ \mathrm{W})\mathrm{Z}) \div \mathrm{Z}$, where $\div$ is the elementwise division, we obtain the form for the randomized shortest path dissimilarity matrix [50, 101],

$$\Delta_{\mathrm{RSP}} = \frac{\mathrm{S} + \mathrm{S}^{\mathrm{T}} - e(\mathrm{diag}(\mathrm{S}))^{\mathrm{T}} - \mathrm{diag}(\mathrm{S})e^{\mathrm{T}}}{2}. \tag{2.57}$$

This is unfortunately not a distance, because it does not verify the triangle inequality. Nonetheless, it is a family of dissimilarity, as it depends on θ and converges to indicate the shortest-path distance when $\theta \to \infty$ while providing half of the commute-time distance when $\theta \to 0^+$. The calculation of Δ_{RSP} utilizes **Algorithm 3.6** of [29].

2.5.7. Bag-of-paths absorption probabilities

The next quantity of interest from the same framework is the probability of absorption when the starting state form a transient state of the network [25, 37, 49, 55, 92]. The first stage is choosing the vertices to investigate - the absorbing nodes. To create these, we have to set the corresponding rows of matrix W to zero such that a random walker cannot go further. The resulting matrix is called W_a.

The likelihood of note now is that of absorption by the node $\alpha \in \mathcal{A}$, knowing that the starting point of the path is $i \in \mathcal{T}$. The symbols $\mathcal{A}$ and $\mathcal{T}$ represent the sets of the absorbing and transient vertices, respectively. By defining the matrix $Z_a = (I - W_a)^{-1}$, the probability in question [55] is indicated by

$$P(e = \alpha | s = i) = \frac{z^a_{i\alpha}}{\sum_{\alpha' \in \mathcal{A}} z^a_{i\alpha'}}. \tag{2.58}$$

The method to compute this likelihood with simple matrix manipulations can be found in **Algorithm 3.7** of [29].

2.5.8. Bag-of-paths covariance measure between nodes

The final measure based on the bag-of-paths framework uses the covariance between the nodes of a graph [67]. The principle is that two vertices are considered highly correlated *if they often co-occur together on the same, preferably short, paths.* The parameter θ is again of significance; the larger its value, the more local the covariance, whereas a small θ denotes consideration for the global structure.

Calculating the covariance between nodes requires taking into account the regular routes. Zero-length paths are excluded; they only involve one vertex and are therefore irrelevant. The process involves some notions; the partition function $\bar{\mathcal{Z}}$ is defined as the denominator of the Gibbs-Boltzmann (equation 2.39). In the case without zero-length paths,

$$\bar{\mathcal{Z}} = z_{\bullet\bullet} - n. \tag{2.59}$$

Next, we need some quantities related to the free energy by standard results of statistical physics [26, 44, 47, 77]. The first is the expected number of times the link $k \to k'$ occurs on a path sampled from a bag-of-paths is equal to the partial derivative of the free energy

$$\bar{n}(k, k') \triangleq \frac{\partial \phi}{\partial c_{kk'}} = -\frac{1}{\theta} \frac{\partial \log \bar{\mathcal{Z}}}{\partial c_{kk'}}. \tag{2.60}$$

This allows us to determine the estimated number of passages through node l as

$$\bar{n}_l \triangleq \sum_{k=1}^{n} \bar{n}(k, l). \tag{2.61}$$

Subsequently, we can obtain the centered, expected instances that $k \to k'$ and $l \to l'$ are traversed together along a path [67] by taking the second-order partial derivative,

$$\bar{n}(k, k'; l, l') \triangleq \frac{1}{\theta^2} \frac{\partial^2 (\log \bar{\mathcal{Z}})}{\partial c_{ll'} \partial c_{kk'}}. \tag{2.62}$$

This value already represents a covariance but between edges. To calculate that between nodes, we have to consider the expected number of incoming transitions by summing the last quantity for

$$\mathrm{cov}(k', l') = \sum_{k,l=1}^{n} \bar{n}(k, k'; l, l'). \tag{2.63}$$

This leads to the bag-of-paths covariance measure [67], defined as

$$k_{kl}^{\mathrm{BoP}} = \mathrm{cov}(k,l) = \sum_{\wp \in \overline{\mathcal{P}}} \mathrm{P}(\wp)(\eta(k \in \wp) - \bar{n}_k)(\eta(l \in \wp) - \bar{n}_l), \tag{2.64}$$

where $\eta(k' \in \wp) = \sum_{k=1}^{n} \eta(k \to k' \in \wp)$ represents the instances where node k' is visited by the path $\wp$.

To obtain the covariance for a graph, we need the first and second-order derivatives of $\bar{\mathcal{Z}}$. Some algebraic manipulations [67] produce

$$n(k,k') = w_{kk'} \frac{e^{\mathrm{T}} Z e_k e_{k'}^{\mathrm{T}} Z e}{\bar{\mathcal{Z}}} = \frac{w_{kk'} z_{\bullet k} z_{k' \bullet}}{\bar{\mathcal{Z}}}, \tag{2.65}$$

where $w_{kk'}$ is an element of W (see equation 2.42), $z_{\bullet k} = \sum_{l=1}^{n} z_{lk}$ and $z_{k\bullet} = \sum_{l=1}^{n} z_{kl}$.

Therefore, the expected number of passages through node k' is

$$\bar{n}_{k'} = \frac{(z_{\bullet k'} - 1) z_{k' \bullet}}{\bar{\mathcal{Z}}}. \tag{2.66}$$

The covariance measure is now only missing the second-order derivative; [67] this yields

$$\bar{n}(k,k';l,l') = w_{kk'} \left\{ \frac{z_{\bullet k} z_{k' \bullet}}{\bar{\mathcal{Z}}} \delta_{kl} \delta_{k'l'} + w_{ll'} \left(\frac{z_{k' \bullet} z_{\bullet l} z_{l'k}}{\bar{\mathcal{Z}}} + \frac{z_{l' \bullet} z_{\bullet k} z_{k'l}}{\bar{\mathcal{Z}}} - \frac{z_{k' \bullet} z_{\bullet k} z_{l' \bullet} z_{\bullet l}}{\bar{\mathcal{Z}}^2} \right) \right\}. \tag{2.67}$$

These equations result in the bag-of-paths covariance measure [67] as follows:

$$\begin{aligned} k_{kl}^{\mathrm{BoP}} = \mathrm{cov}(k,l) \quad = \quad & \frac{1}{\bar{\mathcal{Z}}} \Big\{ (z_{\bullet k} - 1) z_{k\bullet} \delta_{kl} + z_{k\bullet} (z_{\bullet l} - 1)(z_{lk} - \delta_{lk}) \\ & + z_{l\bullet} (z_{\bullet k} - 1)(z_{kl} - \delta_{kl}) - \frac{z_{k\bullet} z_{l\bullet} (z_{\bullet k} - 1)(z_{\bullet l} - 1)}{\bar{\mathcal{Z}}} \Big\}. \end{aligned} \tag{2.68}$$

Algorithm 3.8 of [29] can be used in the computation of this measure.

2.6. Randomized optimal transport on a graph

In the final section of this chapter, I haev described an original method more precisely than its predecessors. This is called *randomized optimal transport on a graph* and has been presented in [38], from which this discussion largerly draws inspiration. It introduces additonal constraint on the starting and ending node distributions of paths to extend the bag-of-paths framework. As its name signalizes, the resulting problem corresponds to a randomized optimal transport on a graph problem [2, 46, 96, 97]. The link between the method and the original graph problem is explained in [38]. In summary, we want to minimize the free energy, subject to margin constraints in order to satisfy the predefined "supply" and "demand" nodes, and uncover the optimal probability on the set of paths.

2.6.1. Definition of the problem

First, I have outlined the problem. The tools used here are the same as in the classic bag-of-paths framework [32, 67], which we have to extend. To this end, two additional density vectors on nodes are introduced, σ_{in} and σ_{out}[4]. These define the new restrictions on the distribution margins of our bag-of-paths probabilities:

$$\mathrm{P}(\mathrm{S}=i) \triangleq \sum_{j\in\mathcal{V}} \sum_{\wp_{ij}\in\mathcal{P}_{ij}} \mathrm{P}(\wp_{ij}) = \sigma_i^{\text{in}} \qquad \forall i \in \mathcal{V}, \tag{2.69}$$

$$\mathrm{P}(\mathrm{E}=j) \triangleq \sum_{i\in\mathcal{V}} \sum_{\wp_{ij}\in\mathcal{P}_{ij}} \mathrm{P}(\wp_{ij}) = \sigma_j^{\text{out}} \qquad \forall j \in \mathcal{V}. \tag{2.70}$$

The above means that the probability of i being the starting point of a path chosen from the bag-of-paths is equal to σ_i^{in}, and the probability that of j being the ending node is σ_j^{out}.

An intuitive interpretation of the problem is as follows [38] : the model assumes that we are carrying a unit of goods from the set of (supply) vertices $\mathcal{I}n = \{i \in \mathcal{V} : \sigma_i^{\text{in}} > 0\}$ to the (demand) set $\mathcal{O}ut = \{j \in \mathcal{V} : \sigma_j^{\text{out}} > 0\}$ by optimizing the balance between expected cost entropy of the paths. Mathematically, it can be written as

$$\begin{aligned} \underset{\{\mathrm{P}(\wp)\}}{\text{minimize}} \quad & \mathrm{FE}(\mathrm{P}) = \sum_{\wp\in\mathcal{P}} \mathrm{P}(\wp)\tilde{c}(\wp) + \mathrm{T}\sum_{\wp\in\mathcal{P}} \mathrm{P}(\wp)\log\left(\frac{\mathrm{P}(\wp)}{\mathrm{P}^{\text{ref}}(\wp)}\right) \\ \text{subject to} \quad & \sum_{j\in\mathcal{V}} \sum_{\wp_{ij}\in\mathcal{P}_{ij}} \mathrm{P}(\wp_{ij}) = \sigma_i^{\text{in}}, \\ & \sum_{i\in\mathcal{V}} \sum_{\wp_{ij}\in\mathcal{P}_{ij}} \mathrm{P}(\wp_{ij}) = \sigma_j^{\text{out}}. \end{aligned} \tag{2.71}$$

In this situation, therefore, the temperature parameter favors the least-cost paths when $\mathrm{T}\to 0$, as well as those with high likelihood $\tilde{\pi}^{\text{ref}}(\wp)$ (equation 2.40) when $\mathrm{T}\to\infty$. We now have to define $\mathrm{P}^{\text{ref}}(\wp)$ according to $\tilde{\pi}^{\text{ref}}(\wp)$, which is not as simple as with the margin constraints.

2.6.2. Computation of the new reference probabilities

First, to ensure the convergence $\mathrm{P}^\star \to \mathrm{P}^{\text{ref}}(\wp)$ in a pure random walk case ($\mathrm{T}\to\infty$), the reference probabilities should have appropriate margins, expressed as

$$\mathrm{P}^{\text{ref}}(\mathrm{S}=i) \triangleq \sum_{j\in\mathcal{V}} \sum_{\wp_{ij}\in\mathcal{P}_{ij}} \mathrm{P}^{\text{ref}}(\wp_{ij}) = \sigma_i^{\text{in}} \qquad \forall i \in \mathcal{V}, \tag{2.72}$$

$$\mathrm{P}^{\text{ref}}(\mathrm{E}=j) \triangleq \sum_{i\in\mathcal{V}} \sum_{\wp_{ij}\in\mathcal{P}_{ij}} \mathrm{P}^{\text{ref}}(\wp_{ij}) = \sigma_j^{\text{out}} \qquad \forall j \in \mathcal{V}, \tag{2.73}$$

$$\sum_{i,j=1}^{n} \mathrm{P}^{\text{ref}}(\wp_{ij}) = 1. \tag{2.74}$$

[4] these two vectors have as conditions $\sum_{i\in\mathcal{V}}\sigma_i^{\text{in}} = \sum_{i\in\mathcal{V}}\sigma_i^{\text{out}} = 1$ and $\sigma_i^{\text{in}}, \sigma_i^{\text{out}} \geq 0, \forall i \in \mathcal{V}$.

If the reference probability matrix is defined in the same manner as in the regular bag-of-paths framework, however, the distribution of the ending node $\mathrm{P}^{\mathrm{ref}}(\mathrm{E}=j)$ would depends solely on the transition matrix of the Markov chain demarcated by $\mathrm{P}^{\mathrm{ref}}$; it would also correspond to the desired distribution σ_{out} only by chance. To fix this problem, a new Markov chain leading to the desired result has to be created.

Therefore, let K_t be a new *killed* Markov process with the substochastic[5] transition matrix $\hat{\mathrm{P}}^{\mathrm{ref}}$ denoted by

$$\hat{\mathrm{P}}^{\mathrm{ref}} \triangleq (\mathrm{I}-\mathrm{Diag}(\alpha))\mathrm{P}^{\mathrm{ref}}, \tag{2.75}$$

where α contains the killing rates of each node, the probability for the path to be killed at i is equal to α_i and $\mathrm{Diag}(\alpha)$ is a diagonal matrix with the elements of the vector α on its diagonal.

Next, we have to match α_i, the probabilities of being killed at node i, with the distribution of the demand vertices σ_{out}. In other words, we need to find K_t following $\hat{\mathrm{P}}^{\mathrm{ref}}$ such that

$$\mathrm{P}(\mathrm{K}_0=i) = \sum_{j\in\mathcal{V}} \mathrm{P}(\mathrm{K}_0=i,\mathrm{K}_{\mathrm{M}-1}=j)=\sigma_i^{\mathrm{in}}, \tag{2.76}$$

$$\mathrm{P}(\mathrm{K}_{\mathrm{M}-1}=j) = \sum_{i\in\mathcal{V}} \mathrm{P}(\mathrm{K}_0=i,\mathrm{K}_{\mathrm{M}-1}=j)=\sigma_j^{\mathrm{out}}, \tag{2.77}$$

in which M is a random variable corresponding to the step where the process is killed. The random walker disappears after reaching the ending node $\mathrm{K}_{\mathrm{M}-1}$. The first constraint is the same as in a standard Markov chain, but to satisfy the second, α has to match σ_{out}. This calculationis detailed in [38] and produces

$$\alpha = \sigma_{\mathrm{out}} \div \bar{n}^{\mathrm{ref}}, \tag{2.78}$$

where $\bar{n}^{\mathrm{ref}}$ contains the expected number of visits to each vertex when following the reference random walk; it is equal to

$$\bar{n}^{\mathrm{ref}} = \left(\mathrm{I}-(\mathrm{P}^{\mathrm{ref}})^{\mathrm{T}}\right)^{+}\left(\sigma_{\mathrm{in}}-(\mathrm{P}^{\mathrm{ref}})^{\mathrm{T}}\sigma_{\mathrm{out}}\right)+\epsilon\pi. \tag{2.79}$$

In the above equation, "+" represents the Moore-Penrose pseudoinverse, π is the stationary distribution of the Markov chain defined by $\mathrm{P}^{\mathrm{ref}}$ and ϵ is an additional free parameter known as *persistence*.

We can now find the probability for such a process of starting at i and being killed at j:

$$\begin{aligned}
\mathrm{P}(\mathrm{K}_0=i,\mathrm{K}_{\mathrm{M}-1}=j) &= \sigma_i^{\mathrm{in}}\left[\sum_{t=0}^{\infty}(\hat{\mathrm{P}}^{\mathrm{ref}})^t\right]_{ij}\alpha_j && (2.80)\\
&= \sigma_i^{\mathrm{in}}\sum_{\wp_{ij}\in\mathcal{P}_{ij}}\hat{\pi}^{\mathrm{ref}}(\wp_{ij})\alpha_j && (2.81)\\
&= \sum_{\wp_{ij}\in\mathcal{P}_{ij}}\mathrm{P}^{\mathrm{ref}}(\wp_{ij}) && (2.82)\\
&= \mathrm{P}(\mathrm{S}=i,\mathrm{E}=j). && (2.83)
\end{aligned}$$

[5] A substochastic matrix is a square matrix with nonnegative entries so that every row adds up to at most 1[69]

This fulfills appropriate margins. The transition from equation 2.80 to 2.81 is due to the convergence of the infinite series, as $\hat{\mathrm{P}}^{\mathrm{ref}}$ is substochastic:

$$\left[\sum_{t=0}^{\infty}(\hat{\mathrm{P}}^{\mathrm{ref}})^t\right]_{ij} = \sum_{\wp_{ij}\in\mathcal{P}_{ij}}\hat{\pi}^{\mathrm{ref}}(\wp_{ij}),$$

where

$$\hat{\pi}^{\mathrm{ref}} \triangleq \prod_{\tau=1}^{t}\hat{p}^{\mathrm{ref}}_{k_{\tau-1}k_\tau}. \tag{2.84}$$

For the conversion between equations 2.81 and 2.82, we simply define

$$\mathrm{P}^{\mathrm{ref}} \triangleq \sigma_i^{\mathrm{in}}\alpha_j\hat{\pi}^{\mathrm{ref}}(\wp_{ij}). \tag{2.85}$$

2.6.3. Optimal probability distribution

The goal of this subsection is to derive the solution for the optimal probability distribution $\mathrm{P}^\star(\wp)$, solving problem 2.71 using Lagrange multipliers. The Lagrange function associated with this problem is

$$\begin{aligned}\mathcal{L}(\mathrm{P},\lambda_{\mathrm{in}},\lambda_{\mathrm{out}}) \triangleq\ & \sum_{\wp\in\mathcal{P}}\mathrm{P}(\wp)\tilde{c}(\wp) + \mathrm{T}\sum_{\wp\in\mathcal{P}}\mathrm{P}(\wp)\log\left(\frac{\mathrm{P}(\wp)}{\mathrm{P}^{\mathrm{ref}}(\wp)}\right)\\ &+\sum_{i\in\mathcal{V}}\lambda_i^{\mathrm{in}}\left[\textstyle\sum_{j\in\mathcal{V}}\sum_{\wp_{ij}\in\mathcal{P}_{ij}}\mathrm{P}(\wp_{ij}) - \sigma_i^{\mathrm{in}}\right]\\ &+\sum_{j\in\mathcal{V}}\lambda_j^{\mathrm{out}}\left[\textstyle\sum_{i\in\mathcal{V}}\sum_{\wp_{ij}\in\mathcal{P}_{ij}}\mathrm{P}(\wp_{ij}) - \sigma_j^{\mathrm{out}}\right],\end{aligned} \tag{2.86}$$

with λ_i^{in} and λ_j^{out} as the Lagrange multipliers. To find the optimal answer, we take the partial derivative with respect to $\mathrm{P}(\wp_{ij})$ and equalize the result to zero. By defining the inverse temperature $\beta \triangleq 1/\mathrm{T}$ and using the equation 2.85, we obtain

$$\mathrm{P}^\star(\wp_{ij}) = \mu_i^{\mathrm{in}}\sigma_i^{\mathrm{in}}\mu_j^{\mathrm{out}}\alpha_j\hat{\pi}^{\mathrm{ref}}(\wp_{ij})\exp(-\beta\tilde{c}(\wp_{ij})), \tag{2.87}$$

where $\mu_i^{\mathrm{in}} \triangleq \exp(-\beta\lambda_i^{\mathrm{in}})$ and $\mu_j^{\mathrm{out}} \triangleq \exp(-\beta\lambda_j^{\mathrm{out}})$. We can now write the likelihood of picking a path from i to j as

$$\begin{aligned}\pi_{ij} &\triangleq \mathrm{P}^\star(\mathrm{S}=i,\mathrm{E}=j) = \sum_{\wp_{ij}\in\mathcal{P}_{ij}}\mathrm{P}^\star(\wp_{ij})\\ &= \mu_i^{\mathrm{in}}\sigma_i^{\mathrm{in}}\mu_j^{\mathrm{out}}\alpha_j\sum_{\wp_{ij}\in\mathcal{P}_{ij}}\hat{\pi}^{\mathrm{ref}}(\wp_{ij})\exp(-\beta\tilde{c}(\wp_{ij})).\end{aligned} \tag{2.88}$$

Although we already have an expression for the optimal probability distribution, we can proceed by defining the fundamental matrix as $\widehat{\mathrm{Z}} \triangleq (\mathrm{I}-\widehat{\mathrm{W}})^{-1}$ with $\widehat{\mathrm{W}} \triangleq \widehat{\mathrm{P}}^{\mathrm{ref}}\circ\exp[-\beta\mathrm{C}]$. It is now possible to demonstrate through a development similar to [32, 67] that $\sum\wp_{ij}\in\mathcal{P}_{ij}\hat{\pi}^{\mathrm{ref}}(\wp_{ij})\exp(-\beta\tilde{c}(\wp_{ij})) = \hat{z}_{ij} = [\widehat{\mathrm{Z}}]_{ij}$ and thus

$$\mathrm{P}^\star(\mathrm{S}=i,\mathrm{E}=j) = \mu_i^{\mathrm{in}}\sigma_i^{\mathrm{in}}\mu_j^{\mathrm{out}}\alpha_j\hat{z}_{ij}. \tag{2.89}$$

This allows us to write a closed-form solution for $\mathit{\Pi} = (\pi_{ij})$,

$$\mathit{\Pi} = \mathrm{Diag}(\mu_{\mathrm{in}}\circ\sigma_{\mathrm{in}})\widehat{\mathrm{Z}}\ \mathrm{Diag}(\mu_{\mathrm{out}}\circ\alpha) \tag{2.90}$$

called the coupling matrix [96, 97].

We have yet to determine the values of the multipliers, which can be deduced from the Lagrangian dual problem:

$$\underset{\lambda_{\text{in}},\lambda_{\text{out}}}{\text{maximize}} \quad \mathcal{L}(\mathrm{P}^\star, \lambda_{\text{in}}\lambda_{\text{out}}). \tag{2.91}$$

Taking the partial derivative with respect to $\lambda_{\text{in}}, \lambda_{\text{out}}$ and using the results from equations 2.87, 2.78 and 2.79, we obtain

$$\mu_{\text{in}} = e \div \left(\widehat{Z}(\mu_{\text{out}} \circ \alpha)\right) \tag{2.92}$$

$$\mu_{\text{out}} = \bar{n}^{\text{ref}} \div \left(\widehat{Z}^{\mathrm{T}}(\mu_{\text{in}} \circ \sigma_{\text{in}})\right). \tag{2.93}$$

These two quantities are computed in iteratively until convergence by fixing an arbitrary $\mu_{\text{out}}^{(0)}$. The reader may notice that this section only considers non-hitting paths up to this point, but a version of these probabilities for the other case exists. This is the subject of the next segment.

2.6.4. The bag-of-hitting-paths case

This new scenario adds the margin constraints to the hitting-paths framework developed in section 2.5.3. Now, we have to find the optimal hitting paths probability distribution, $\mathrm{P}_h^\star(\wp)$, solving

$$\begin{aligned} \underset{\{\mathrm{P}_h(\wp)\}}{\text{minimize}} \quad & \mathrm{FE}_h(\mathrm{P}) = \sum_{\wp \in \mathcal{P}^h} \mathrm{P}_h(\wp)\tilde{c}(\wp) + \mathrm{T} \sum_{\wp \in \mathcal{P}^h} \mathrm{P}_h(\wp) \log\left(\frac{\mathrm{P}_h(\wp)}{\mathrm{P}_h^{\text{ref}}(\wp)}\right) \\ \text{subject to} \quad & \sum_{j \in \mathcal{V}} \sum_{\wp_{ij} \in \mathcal{P}_{ij}^h} \mathrm{P}_h(\wp_{ij}) = \sigma_i^{\text{in}}, \\ & \sum_{i \in \mathcal{V}} \sum_{\wp_{ij} \in \mathcal{P}_{ij}^h} \mathrm{P}_h(\wp_{ij}) = \sigma_j^{\text{out}}. \end{aligned} \tag{2.94}$$

We notice the two problems are equivalent as likelihood for paths containing the final nodes several times tends toward zero in the non-hitting case for $\mathrm{T} \to 0$. However, this is not the case when $\mathrm{T} \to \infty$, due to the difference between the two models reference probabilities and the structures of the paths.

The main variation between the two problems remains the reference likelihoods $\mathrm{P}_h^{\text{ref}}$, which are much simpler in this case. By recalling that the sum of all $\wp \in \mathcal{P}_{ij}^h$ of $\tilde{\pi}^{\text{ref}}(\wp) \triangleq \prod_{\tau=1}^{t} p^{\text{ref}}_{i_{\tau-1},i_\tau}$ is equal to one [32], we can easily geneate

$$\mathrm{P}_h^{\text{ref}}(\wp_{ij}) \triangleq \sigma_i^{\text{in}} \sigma_j^{\text{out}} \tilde{\pi}^{\text{ref}}(\wp_{ij}). \tag{2.95}$$

The same reasoning as in the previous section yields

$$\mathrm{P}^\star(\wp_{ij}) = \mu_i^{\text{h,in}} \sigma_i^{\text{in}} \mu_j^{\text{h,out}} \sigma_j^{\text{out}} \tilde{\pi}^{\text{ref}}(\wp_{ij}) \exp(-\beta \tilde{c}(\wp_{ij})), \tag{2.96}$$

with $\mu_i^{\mathrm{h,in}} \triangleq \exp(-\beta\lambda_i^{\mathrm{h,in}})$ and $\mu_i^{\mathrm{h,out}} \triangleq \exp(-\beta\lambda_i^{\mathrm{h,out}})$. The closed-form can be written as

$$\pi_{ij}^h \triangleq \mathrm{P}^\star(\mathrm{S}=i,\mathrm{E}=j) = \mu_i^{\mathrm{h,in}}\sigma_i^{\mathrm{in}}\mu_j^{\mathrm{h,out}}\sigma_j^{\mathrm{out}}z_{ij}^h, \tag{2.97}$$

and we can so obtain the coupling matrix,

$$\Pi_h = \mathrm{Diag}(\mu_{\mathrm{in}}^h \circ \sigma_{\mathrm{in}})\widehat{\mathrm{Z}}_h\ \mathrm{Diag}(\mu_{\mathrm{out}}^h \circ \sigma_{\mathrm{out}}), \tag{2.98}$$

in which μ_{in}^h and μ_{out}^h are determined iteratively in the dual space according to

$$\mu_{\mathrm{in}}^h = e \div \left(\widehat{\mathrm{Z}}_h(\mu_{\mathrm{out}}^h \circ \sigma_{\mathrm{out}})\right) \tag{2.99}$$

$$\mu_{\mathrm{out}}^h = \bar{n}^{\mathrm{ref}} \div \left(\widehat{\mathrm{Z}}_h^{\mathrm{T}}(\mu_{\mathrm{in}}^h \circ \sigma_{\mathrm{in}})\right). \tag{2.100}$$

2.6.5. Derived distances

We can now identify a distance known as the surprisal distance, and, generalizing the one introduced in 2.5.4 with the particularity, we affect the results through σ_{in} and σ_{out}. The constrained bag-of-paths surprisal distance is defined by

$$\Delta_{ij}^{\mathrm{sur}} = \begin{cases} -\frac{1}{2}\left(\log(\pi_{ij}) + \log(\pi_{ji})\right) & \text{if } i \neq j \\ 0 & \text{if } i = j, \end{cases}$$

and its bag-of-hitting-paths variant by

$$\Delta_{ij}^{\mathrm{h,sur}} = \begin{cases} -\frac{1}{2}\left(\log(\pi_{ij}^h) + \log(\pi_{ji}^h)\right) & \text{if } i \neq j \\ 0 & \text{if } i = j. \end{cases}$$

Now, many measures to compute the (dis)similarity have been defined, we need to test them. In this end, a semi-supervised classification task is used. The next chapter explains this classification task as well as the statistical tests used to compare them.

Experimental methodology 3

Contents

3.1. Preliminaries

After the description of many measures in the previous chapter, this chapter describes the methodology utilized to obtain results and performs comparisons to deduce the most appropriate approaches. First, a description of what semi-supervised classifications represent will be made. Second, a complete explanation of the experimental procedure is developed. Next, I tackle the various statistical tools used to interpret the results. Finally, I briefly discuss the datasets on which the methods were tested.

Moreover, from this section onward, some abbreviations are used in place of the complete names of the (dis)similarity measures. These are listed in Table 3.1. I have added the suffixes *-mds* and *-g* corresponding, respectively, to denote multidimensional scaling and Gaussian mapping (Sections 2.3.1 and 2.3.2).

3.2. Semi-supervised classification

We will now discuss briefly the semi-supervised classification concept [90, 103]. The semi-supervised classification is part of *supervised learning* which is the most popular paradigm in machine learning [90]. The aim of this paradigm is to learn from a limited amount of

labeled data. According to the amount on supervisory information contained in the labeled data, we talk about *unsupervised learning*, *supervised learning* or *semi-supervised learning*. Here, we will focus on this last case.

In *semi-supervised learning*, only a small part of the dataset $\mathcal{D}$ is labeled while the large part is left unlabeled. It allows to combine the benefits from both labeled and unlabeled data. If we define $\mathcal{X}$ as the set of inputs and $\mathcal{Y}$ as the set of possible outputs, a *semi-supervised learning* algorithm has to learn a function $f : \mathcal{X} \to \mathcal{Y}$ from a predefined family of functions $\mathcal{F}$ given a dataset $\mathcal{D} = \{\mathcal{D}_l, \mathcal{D}_u\}$ where $\mathcal{D}_l = \{(x_i, y_i)\}_{i=1}^{n_l}$ is the set of labeled data and $\mathcal{D}_l = \{x_i\}_{i=1}^{n_u}$ is the set of unlabeled ones. We may add that, in most of cases, the amount of labeled data is much smaller than the one of unlabeled data, $n_u \gg n_l$.

It exists two categories of *semi-supervised learning* algorithms according to the nature of the output domain $\mathcal{Y}$. If it is a continuous domain, the algorithm is referred to as *regression* while if the domain is discrete, the algorithm is called *classification*. Our algorithm will be part of this last category.

In the context of graphs, a data sample is represented by a node of a weighted graph where the weights associated to the edges providing a measure of similarity between nodes. By combining the information given by the graph structure and the labels of a subset of the nodes, a *semi-supervised* algorithm tries to classify the remainder of the nodes. Now, we have described the generalities of *semi-supervised* algorithms, we will develop, in the next section, the algorithm used in this master thesis, the sum of similarities.

3.3. General description of the methodology

A summary of the techniques used is provided here. After performing a randomized permutation of the adjacency matrix to avoid originally structured datasets, the method generates five splits of the vector containing the labels for each node. This vector has been provided in the dataset. The splits enable external cross-validation, which consists of following the rest of the procedure several times to take the final mean and standard deviation of the numbers obtained. Averaging several experiments eliminates calculation error and thus yields results in which we can be more confident.

Each split is composed of two vectors. The first is called the *training* vector; it contains the indices of approximately 20% of the nodes, for which we keep the real class labels. The second is called the *test* vector; it involves the other 80%, for which we omit the tags. The goal is to measure a similarities matrix from the adjacency matrix and recover the real class labels of the maximum node of the *test* vector using the sum of similarities algorithm described in the next section. There are five splits, each with a *training* vector containing approximately 20% of the vertices. This is not coincidental and the calculation is performed

Abbreviation	**Complete name**
KS	Katz Similarity
LE	Leicht's extension
CTD	Commute-time distance
ECTD	Euclidean commute-time distance
CCTD	Corrected commute-time distance
CECTD	Corrected Euclidean commute-time distance
SR	SimRank similarity
EDK	Exponential diffusion kernel
LDK	Laplacian exponential diffusion kernel
MRLK	Modified regularized Laplacian kernel
CTK	Commute-time kernel
CCTK	Corrected commute-time kernel
RCTK	Regularized commute-time kernel
RWWR	Random walk with restart similarity
MDSD	Markov diffusion square distance
TAMSD	Time-averaged Markov diffusion squared distance
MDK	Markov diffusion kernel
TAMDK	Time-averaged diffusion kernel
LFD	Logarithmic forest distance
BoP	Regular bag-of-paths
BoHPwZP	bag-of-hitting-paths with zero-length paths
BoHP	bag-of-hitting-paths without zero-length paths
BoHPSD	bag-of-hitting-paths surprisal distance
FE	Free energy distance
RSPD	Randomized shortest path dissimilarity
BoPAP	Bag-of-paths absorption probabilities
BoPC	Bag-of-paths covariance measure
ROTwHP	Random optimal transport with hitting paths
ROT	Random optimal transport without hitting paths

Table 3.1.: Abbreviations for the 29 similarity/dissimilarity measures.

such that each node is only in the *training* vector once.

From now on, every step is externally cross-validated and therefore repeated for each split. First, I verify that each label is located in the *training* set. If not, the split is rejected, because it is impossible to find the missing class label for the node in the *test* set, and the percentage of correct labels found is therefore largely biased for that run. Next, I attempt to the optimal parameter for the (dis)similarity measure and the split with an internal cross-validation which is detailed in Section 3.3.2.

The last stages consist of computing the measure that we want to evaluate. If necessary, we may transform this into a kernel matrix before determining the accuracy of this value for the corresponding split with the sum of similarities. The figure obtained for each split is then recorded, and the mean acrossfive runs is taken to produce the final result for a specific dataset.

3.3.1. Sum of similarities

To test the accuracy of a measure, I choose to use a sum of similarities algorithm [66], the equivalent of a k-nearest neighbors when using distances. This takes similarity (and more precisely a kernel) as a main entry and outputs the percentage of nodes whose process has found the right label. A pseudo-code for this case can be found at algorithm 1.

Algorithm 1 The sum-of-similarities method using a kernel

Input :

- The $n \times n$ kernel matrix K representing similarities between nodes.
- The $n \times 1$ column vector y containing the real class labels of the nodes.
- The *training* vector containing indices of the training nodes set.
- The *test* vector containing indices of the test nodes set.

Output :

- The classification accuracy ac in percent.

$nbc \leftarrow \max(y)$
$yt \leftarrow \text{zeros}(n, 1)$
$yt(\text{training}) \leftarrow y(\text{training})$
for c = 1:nbc **do**
    $\text{Y1} \leftarrow [\text{Y1} \;\; (yt == c)]$
end for
$\text{Y}p \leftarrow \text{K} * \text{Y1}$
$yp \leftarrow \text{index}(\max(\text{Y}p))$
$ac \leftarrow 100 * \dfrac{\text{sum}(yp(test) == y(test))}{\text{length}(y(test))}$
return ac

Detailing the steps of this algorithm, we observe that lines 2 and 3 create a $n \times 1$ vector column in which the i^{th} element is equal to the real class of the corresponding node if it is in the training sample. If not, the vertex belongs to a dummy class 0.

The next three lines generate a $n \times nbc$ matrix, where nbc is the number of classes. Each row of this binary matrix corresponds to a node; if the i^{th} is labeled, the corresponding row has a 1 in the column corresponding to its class and zeroes elsewhere. On the other hand, if it is not labeled, the respective row has zeroes everywhere.

In the seventh line, we compute the sum of similarities well said. Recall that the kernel matrix K contains the similarities between the nodes; thus, if it is multiplied by previous created matrix, an element of the resulting matrix is equal to the sum of similarities between a vertex and the labeled nodes corresponding to a certain class. Therefore, we obtain a new $n \times nbc$ matrix, y_p, where each row matches a vertex, and each column, a class. The (i^{th}, j^{th}) element of this matrix is equal to the sum of similarities between the i^{th} node and those contained in the training sample, which are labeled with class j. Mathematically, we have in matrix form

$$y_p = \mathrm{K} \times y \tag{3.1}$$

where y is the matrix with the known labels.

Finally, the eighth line assigns the vertices to the class of maximum similarity, before the last line computes the percentage of originally unlabeled nodes for which this method has identified the correct class.

3.3.2. Parameter tuning

As noted in chapter 2, most of the (dis)similarity measures depend on parameters. For each experiment, we have to find the value that yields the most accurate result. This is obviously different for each method, but it also varies between datasets and the various splits within them. To obtain this optimal figure, we have to test a finite range of arbitrarily chosen values for the parameters with an internal cross-validation similar to the external one described before. The selection depends on the approach used and an overview is provided in Table 3.2.

We have yet to explain how the internal cross-validation generates the most ideal value for the parameter. This process operates in a varying order from the external. In this case, we first calculate the evaluated similarity and then perform the cross-validation with a smaller vector to accelerate the overall computation. Instead of splits represented by *training* vectors containing 20% of the nodes and *test* ones with the other 80%, the internal cross-validation creates new splits by referencing the *training* vectors of the external process. Thus, for each value within the range, we take the mean of five sums of similarities[1] that, as a new *training*

[1] Again, the test is done only if each class are represented in the internal *training* set

Measure	Parameter	Range of values
KS	$0 < \alpha \leq 1/\rho(A) \triangleq s$	$s/1000, s/100, s/10, s/5, s/2, s$
LE	$0 < \alpha \leq 1/\rho(A) \triangleq s$	$s/1000, s/100, s/10, s/5, s/2, s$
SR	$\alpha \in]0,1[$	0.01, 0.1, 0.3, 0.5, 0.7, 0.9, 0.99
EDK	$\alpha > 0$	0.01, 0.1, 0.5, 1, 5, 10
LDK	$\alpha > 0$	0.01, 0.1, 0.5, 1, 5, 10
MRLK	$\alpha > 0$	0.01, 0.1, 0.5, 1, 5, 10
	$0 \leq \gamma \leq 1$	0.01, 0.1, 0.3, 0.5, 0.7, 0.9, 0.99
RCTK	$\alpha \in]0,1[$	0.01, 0.1, 0.3, 0.5, 0.7, 0.9, 0.99
RWWR	$\alpha \in]0,1[$	0.01, 0.1, 0.3, 0.5, 0.7, 0.9, 0.99
LFD	$\alpha > 0$	0.01, 0.1, 0.5, 1, 5, 10
BoP	$\theta > 0$	0.01, 0.1, 0.5, 1, 5, 10
BoHPwZP	$\theta > 0$	0.01, 0.1, 0.5, 1, 5, 10
BoHP	$\theta > 0$	0.01, 0.1, 0.5, 1, 5, 10
BoHPSD	$\theta > 0$	0.01, 0.1, 0.5, 1, 5, 10
FE	$\theta > 0$	0.01, 0.1, 0.5, 1, 5, 10
RSPD	$\theta > 0$	0.01, 0.1, 0.5, 1, 5, 10
BoPAP	$\theta > 0$	0.01, 0.1, 0.5, 1, 5, 10
BoPC	$\theta > 0$	0.01, 0.1, 0.5, 1, 5, 10
ROTwHP	$\beta > 0$	0.01, 0.1, 0.5, 1, 5, 10
ROT	$\beta > 0$	0.01, 0.1, 0.5, 1, 5, 10

Table 3.2.: Range of tested values for the parameters of the parametric measures.

set, 20% of the nodes present in the original one (or 4% of the total number of nodes). The internal *test* vector consists of the remaining 80%. After five such tests with different sets, the internal cross-validation returns the parameter with the most ideal mean result.

3.4. Comparison metrics and tools

Now that the procedure has been described, we know how we will obtain results. Nevertheless, we do not know yet how they will be compared. Therefore, the following section will include a description of some useful tools to help us for this task.

3.4.1. Borda Ranking

The Borda ranking is originally a weighted voting system formalized by Borda in [21]. Adapting it for this situation generates a score for each measure tested. To this end, we examine the accuracy thereof for a dataset and provide a number of points for each that corresponds to its result. In other words, if we test n different measures, the one that yields the best outcomes receives n points, the second has $n-1$ and so on until the last, which obtains only one point. We repeat this for every dataset, and the final score of a measure is simply the sum of the points obtained.

To improve this tool, the possibility of a tie is included. If two measures have results whose difference is smaller than ε, then they would have the same score, and the subsequent measure would be evaluated as if there had been no ties.

3.4.2. Friedman-Nemenyi test

Imagining that the datasets could lead to totally different results according to their features, the utilization of simple statistical tools as the mean would conduct to irrelevant comparisons. However, some tests as Friedman-Nemenyi's allows to understand if it exists a real difference between several methods.

We begin the description with the Friedman test [33, 34], a statistical assessment that is a non-parametric equivalent of the repeated-measures ANOVA. For each individual dataset, it ranks the algorithms as follows: one for best-performing algorithm, two for the second best, and so on. If two procedures lead to the same results, both would receive the average grade.

If r_i^j is the rank of the j^{th} out of k algorithms for the i^{th} of N datasets, [22] sates the null hypothesis of this test: that all the processes are equivalent, and thus their mean ranks $R_j = \frac{1}{N}\sum_i r_i^j$ should be equal at a confidence level of α. Under this postulation, the Friedman statistic

$$\chi_F^2 = \frac{12N}{k(k+1)}\left[\sum_j R_j^2 - \frac{k(k+1)^2}{4}\right] \tag{3.2}$$

is distributed according to the chi-square distribution χ_F^2 with $k-1$ degrees of freedom if N and k are sufficiently large (typically at values of $N > 10$ and $k > 5$). It is possible to proceed as in [39], which claims that the Friedman statistic can be improve with a better statistic

$$F_F = \frac{(N-1)\chi_F^2}{N(k-1)-\chi_F^2}, \tag{3.3}$$

which uses the F-distribution with $k-1$ and $(k-1)(N-1)$ degrees of freedom; the table of critical values can be obtained from the associated tables in any statistical book.

To check if the null hypothesis is accepted, we simply compute F_F and compare it with the critical value $V_{crit} \triangleq F(k-1,(k-1)(N-1))$ from a reference book for the chosen level of confidence. If $F_F > V_{crit}$, we reject the null hypothesis; thus, the average measured ranks are significantly different from the mean.

If it is the case, a post-hoc evaluation such as the Nemenyi [72] test, in which all algorithms are compared to each other, can be adopted. This allows us to declare that two procedures are significantly different if the corresponding average ranks diverge by at least

$$CD = q_\alpha \sqrt{\frac{k(k+1)}{6N}} \tag{3.4}$$

Number of algorithms	11	12	13	14	15	16	17	18	19	20
$q_{0.01}$	3.696	3.741	3.781	3.818	3.853	3.884	3.914	3.941	3.967	3.991
$q_{0.05}$	3.219	3.268	3.312	3.354	3.391	3.426	3.458	3.488	3.517	3.544
$q_{0.10}$	2.978	3.030	3.077	3.120	3.159	3.196	3.230	3.261	3.291	3.319
Number of algorithms	41	42	43	44	45	46	47	48	49	50
$q_{0.01}$	4.318	4.329	4.339	4.349	4.359	4.368	4.378	4.389	4.395	4.404
$q_{0.05}$	3.899	3.911	3.922	3.933	3.943	3.954	3.964	3.973	3.983	3.992
$q_{0.10}$	3.691	3.703	3.714	3.726	3.737	3.747	3.758	3.768	3.778	3.788

Table 3.3.: Range of critical values for the Nemenyi test that could interested us available at http://kourentzes.com/forecasting/2014/05/01/critical-values-for-the-nemenyi-test/.

where the critical values for q_α are based on the Studentized range statistic divided by $\sqrt{2}$ [22]. These can be found in Table 3.3.

When multiple algorithms are compared, the results of the Nemenyi test can be visually represented. In this simple diagram, the mean ranks are denoted by a dot and the critical differences by a line that is bisected by the dot. The best processes appear to the right of the axis, and two algorithms are significantly diverse if the dot for one of them misses the line of the other.

3.4.3. Wilcoxon signed-rank test

To improve the comparisons between measures, I have also conducted some pairwise comparisons through a non-parametric technique called the one-sided Wilcoxon signed-rank test [100]. It determines if an algorithm is significantly better than another by comparing their results on all the datasets at a confidence level α. For the original test, we include the possibility of a tie if two processes have outcomes whose difference is smaller than ε.

The evaluation is as follows. Assuming N data points for each similarity measure, $x_{1,i}$ and $x_{2,i}$ are the quantities for all $i = 1, \ldots, N$. We first calculate the absolute difference $|x_{2,i} - x_{1,i}|$ and the sign of the variance $\mathrm{sgn}(x_{2,i} - x_{1,i})$ between the points.

Next, we exclude the pairs with $|x_{2,i} - x_{1,i}| < \varepsilon$ and define N_r as the length of the remaining vector. The differences in absolute values are then sorted, and we rank each of them, starting with one for the smallest difference, two for the second smallest, and so on until the final (largest) designation N_r. If several differences are equal to one another, they are assigned the average rank. The ratings are stocked in a vector R.

It remains to compute the statistics

$$W = \sum_{i=1}^{N_r} \left[\mathrm{sgn}(x_{2,i} - x_{1,i}) \cdot R_i \right]. \tag{3.5}$$

N_r	6	7	8	9	10	11	12	13	14	15
$\alpha = 0.05$	0	2	4	6	8	11	14	17	21	25
$\alpha = 0.02$	-	0	2	3	5	7	10	13	16	20
$\alpha = 0.01$	-	-	0	2	3	5	7	10	13	16

Table 3.4.: Range of critical values for Wilcoxon signed-rank test that could interested us available at http://users.sussex.ac.uk/~grahamh/RM1web/WilcoxonTable2005.pdf.

This result is compared to a critical figure displayed in Table 3.4. The Wilcoxon signed-rank test declares two algorithms significantly different if the absolute value of the statistic is larger than the critical one,

$$|W| > W_{\text{critical},N_r}. \tag{3.6}$$

Enough tools to make helpful observation have been described. However, we still have no idea what the data on which the procedure would perform in order to compare the measures are. The next section should answer to this question.

3.5. Description of the datasets

This section describes the datasets on which I have tested the measures discussed in Chapter 2. These are taken from [32]; each represents a real-world graph and is therefore not randomly generated. Recall that each set consists of the adjacency matrix, as well as a vector containing the labels of the different nodes. An overview is provided in Table 3.5.

3.5.1. Newsgroup datasets

The nine datasets used here come from a global source[2] that contains 20,000 text documents taken from 20 discussion groups on the Usenet diffusion list. The selected materials span various topics, each of which covers nearly 200 documents extracted randomly from the multiple newsgroups [32]. The first three datasets are composed of nearly 400 texts on two issues. Similarly, the next three comprise three classes and nearly 600 documents, while the last three have nearly 1000 documents because they discuss five subjects. The weight of the edges contained in the adjacency matrix represents the terms shared by the texts.

3.5.2. The Internet Movie Database (IMDB)

The collaborative Internet Movie Database [64] is a well-known database of movies. It has several applications, such as making recommendations, clustering or classifying films by category. The dataset can be represented by a graph, and the nodes are the movies, which are linked if they share the same production company. The weight of an edge represents

[2]Available at http://people.csail.mit.edu/jrennie/20Newsgroups/.

the number of such connections. A film is categorized into one of two clusters based on its popularity (whether it is a box-office hit or not).

3.5.3. WebKD datasets

Each of these represents a network, in which a node is the web page within a set gathered from a computer science department. There are four WebKD datasets for four different universities [64]. Each page is manually assigned to one of six categories : course, department, faculty, project, staff and student. Two vertices are linked if their corresponding pages are co-cited (if x links to z and y links to z, then x and y are co-citing z). The weight of the edge between x and y is the product of the sums of the number of hyperlinks from x to z and from y to z.

In this chapter, the complete procedure, the tools and the datasets have been discussed. We can so get an idea of where the results exposed in the next chapter come. We have also the ways to understand the comparisons made with the tests previously described. Therefore, we will now tackle this comparisons and try to make helpful observations.

Table 3.5.: Overview of datasets used to test our measures. For each dataset, we give its size, the names of the different classes as well as their sizes.

Dataset	size	Dataset	size	Dataset	size
news-2cl-1	**400**	**news-2cl-2**	**398**	**news-2cl-3**	**399**
Politics/general	200	Computer/graphics	198	Space/general	200
Sport/baseball	200	Motor/motorcycles	200	Politics/mideast	199
news-3cl-1	**600**	**news-3cl-2**	**598**	**news-3cl-3**	**595**
Sport/baseball	200	Computer/windows	200	Sport/hockey	197
Space/general	200	Motor/autos	198	Religion/atheism	200
Politics/mideast	200	Religion/general	200	Medicine/general	198
news-5cl-1	**998**	**news-5cl-2**	**999**	**news-5cl-3**	**997**
Computer/windows	200	Computer/graphics	200	Computer/mac-hardware	200
Cryptography/general	200	Computer/pc-hardware	200	Sport/hockey	200
Politics/mideast	200	Motor/autos	199	Medicine/general	197
Politics/guns	200	Religion/atheism	200	Religion/general	200
Religion/Christian	198	Politics/mideast	200	Forsale/general	200

Dataset	size
IMDB	**1169**
Box-office hit	597
Non box-office hit	572

Dataset	size	Dataset	size	Dataset	size
WebKB-Cornell	**346**	**WebKB-Texas**	**334**	**WebKB-Washington**	**434**
Course	54	Course	50	Course	151
Department	54	Department	51	Department	170
Faculty	145	Faculty	36	Faculty	39
Project	62	Project	163	Project	44
Staff	25	Staff	28	Staff	20
Student	6	Student	6	Student	10
WebKB-Wisconsin	**348**				
Course	155				
Department	83				
Faculty	37				
Project	37				
Staff	25				
Student	11				

Results and discussions

4

Contents

4.1. Preliminaries

In this chapter, I discuss the results obtained through testing my experiment with various measures. As previously mentioned, the outcomes are the average classification accuracies in percentages for 20% of the known labeled nodes and others for which we are attempting to find the correct label tag. The result is an average, as the investigation was conducted five times with different known labeled vertices for each.

The breakdown of this chapter is as follows. In the first part, I consider the similarity or dissimilarity measures presented in [29], referred to henceforth as *the classical measures*. I compare them and to determine the best option with all the statistical tools available. The second segment addresses the original method introduced in Section 2.6 and where it is located among the measures form the first part.

Note that all the statistical tests here are performed at a significance level of 95%. That means taking an α equal to 0.05.

4.2. Detailed results of the classical measures

All the results obtained can be found in tables 4.1, 4.2, 4.3 and 4.4. Each number represents the accuracy in percent. As the results are expressed as a percentage, the larger the result, the better. For each dataset, the top-performing measure is presented in boldface.

Measure →	KS	KS	LE	LE	CTD	CTD	ECTD	ECTD	CCTD	CCTD	CECT	CECT
Dataset ↓	-g	-mds	-g	-mds	-g	-mds	-g	-mds	-g	-mds	D-g	D-mds
texas	48.80	41.61	48.80	48.80	49.03	7.86	48.95	66.99	48.80	35.10	18.39	34.20
washington	39.17	52.65	39.17	39.17	35.08	41.88	34.79	43.49	39.17	25.53	25.24	28.86
wisconsin	44.54	32.17	44.54	44.54	44.97	48.87	44.54	68.90	44.54	36.99	39.43	35.99
cornell	41.91	42.99	41.91	41.91	41.40	26.02	41.91	42.71	41.91	31.49	35.55	23.62
imdb	51.33	49.91	51.35	51.07	51.01	50.15	50.58	50.77	50.88	50.32	51.05	50.88
news-2cl-1	48.94	45.44	50.00	50.00	50.44	85.19	50.13	95.19	49.94	50.56	50.00	50.19
news-2cl-2	50.63	78.71	50.00	66.15	55.53	87.69	51.82	91.96	50.69	50.06	49.87	50.19
news-2cl-3	48.68	46.81	50.13	50.00	50.19	92.23	50.13	95.86	49.94	50.00	49.19	50.38
news-3cl-1	33.63	45.17	33.33	62.21	33.33	78.38	33.33	92.71	33.29	33.67	33.29	33.38
news-3cl-2	34.41	39.93	33.44	67.68	33.78	87.37	33.49	91.97	33.49	33.40	34.41	33.99
news-3cl-3	32.98	37.82	33.45	33.32	33.78	71.51	33.45	89.58	33.53	33.03	33.24	33.87
news-5cl-1	19.94	29.03	20.04	20.04	20.19	58.60	20.17	87.73	20.24	20.12	19.39	19.91
news-5cl-2	18.27	29.58	20.02	20.02	20.20	61.38	20.02	83.46	20.30	19.57	20.25	20.00
news-5cl-3	18.76	26.23	20.56	20.06	25.42	61.93	24.07	81.69	19.76	20.01	20.01	19.86

Table 4.1.: First part of the classification accuracy in percent for the classical measures, obtained on each dataset and by taking the average of 5 experiments.

Measure →	SR	SR	EDK	LDK	MRLK	CTK	CCTK	RCTK	RWWR	MDSD	MDSD
Dataset ↓	-g	-mds								-mds	-g
texas	48.80	26.26	62.20	68.19	72.08	67.96	62.82	74.63	70.58	48.43	28.60
washington	39.17	36.87	47.12	62.84	52.65	43.84	53.92	67.34	61.92	39.00	25.82
wisconsin	44.54	45.91	63.86	72.34	61.99	69.04	74.00	73.49	72.77	43.68	26.65
cornell	41.91	39.45	49.06	54.77	43.86	44.66	58.02	58.67	53.90	41.91	21.82
imdb	51.67	55.79	78.38	78.74	65.18	49.08	50.09	**79.11**	78.61	51.07	49.44
news-2cl-1	49.88	50.81	94.44	94.94	90.75	95.19	97.00	93.63	94.69	47.69	51.25
news-2cl-2	50.25	51.45	90.95	91.90	92.21	91.96	**93.03**	90.64	91.08	50.63	48.37
news-2cl-3	49.87	49.69	94.11	94.61	95.68	95.86	95.86	94.67	94.49	52.07	49.62
news-3cl-1	33.33	35.13	91.25	92.96	76.25	92.71	**94.25**	92.54	91.29	33.71	31.96
news-3cl-2	33.11	33.78	90.18	91.93	70.52	91.97	**93.77**	92.31	90.34	33.03	32.44
news-3cl-3	31.72	33.82	88.70	89.29	68.49	89.58	91.09	91.85	88.87	32.82	32.18
news-5cl-1	20.04	20.37	86.55	87.68	72.45	87.73	89.38	88.25	86.70	19.14	19.64
news-5cl-2	19.97	20.15	81.56	83.16	40.94	83.46	**84.83**	84.03	81.83	19.59	18.97
news-5cl-3	20.59	19.86	80.54	81.80	40.30	81.69	**84.83**	82.50	80.72	18.98	20.61

Table 4.2.: Second part of the classification accuracy in percent for the classical measures, obtained on each dataset and by taking the average of 5 experiments.

Measure → Dataset ↓	TAMS D-g	TAMS D-mds	MDK	TAM DK	LFD -g	LFD -mds	BoP -g	BoP -mds	BoHP wZP-g	BoHPw ZP-mds	BoHP -g
texas	48.73	36.60	28.53	19.32	48.80	78.07	48.80	14.97	48.80	48.80	48.80
washington	39.00	27.59	28.40	22.87	46.55	67.91	39.17	39.17	39.17	39.17	39.17
wisconsin	43.75	22.63	24.71	24.50	45.12	**74.28**	44.54	44.54	44.54	44.54	44.54
cornell	41.33	20.45	27.03	22.33	41.91	58.82	41.91	15.61	41.91	41.91	41.91
imdb	51.20	49.44	49.62	50.51	50.90	50.17	51.07	51.07	51.07	51.07	51.13
news-2cl-1	49.19	48.88	49.00	48.00	95.25	95.50	48.56	50.00	47.63	50.31	46.13
news-2cl-2	51.32	52.14	50.88	51.38	90.14	91.52	49.81	49.75	50.44	48.93	47.86
news-2cl-3	51.69	53.63	53.57	53.95	93.55	95.74	50.19	50.13	50.19	50.38	44.67
news-3cl-1	33.63	33.00	32.54	33.79	89.67	93.13	32.88	33.58	31.71	33.33	30.38
news-3cl-2	33.57	34.41	32.69	34.57	86.83	92.06	33.53	33.65	33.70	33.07	27.76
news-3cl-3	33.61	33.36	30.92	34.16	78.66	91.18	33.07	33.45	32.69	33.40	30.17
news-5cl-1	20.37	21.39	19.34	21.74	81.74	87.88	22.54	20.04	21.84	20.12	20.77
news-5cl-2	20.22	19.72	19.32	19.84	76.53	83.51	19.12	20.10	18.67	19.97	21.97
news-5cl-3	20.64	22.02	20.91	22.84	76.38	82.82	20.46	20.21	21.21	19.98	19.03

Table 4.3.: Third part of the classification accuracy in percent for the classical measures, obtained on each dataset and by taking the average of 5 experiments.

Measure → Dataset ↓	BoHP -mds	BoHP SD-g	BoHP SD-mds	FE -g	FE -mds	RSPD -g	RSPD -mds	BoP AP-g	BoP AP-mds	BoP C-g	BoP C-mds
texas	51.12	48.80	78.30	48.80	**79.49**	49.70	46.72	48.80	48.80	48.80	52.17
washington	37.96	40.67	43.67	66.53	**69.01**	38.42	27.31	39.17	34.79	41.24	38.54
wisconsin	58.48	44.54	54.74	46.77	74.07	49.00	47.85	44.54	44.54	44.54	48.56
cornell	42.05	41.91	**62.72**	41.91	62.28	42.49	17.85	41.91	41.91	41.91	41.91
imdb	54.75	50.98	50.58	56.13	50.21	50.60	48.46	51.07	51.07	51.07	60.59
news-2cl-1	62.50	94.25	**95.81**	94.69	95.69	93.88	87.75	48.56	51.56	51.19	50.44
news-2cl-2	48.99	92.84	92.65	92.90	92.90	88.06	85.81	49.75	50.00	49.56	52.88
news-2cl-3	67.79	**96.93**	96.43	96.37	96.12	92.36	83.91	50.50	50.13	51.63	51.69
news-3cl-1	31.58	93.75	93.92	93.92	93.54	87.00	82.46	33.38	34.54	35.46	33.67
news-3cl-2	30.81	91.22	91.52	93.06	93.31	86.92	70.64	32.32	33.40	35.82	33.99
news-3cl-3	54.33	89.12	91.93	91.97	**92.48**	71.43	85.80	32.39	33.95	32.10	33.45
news-5cl-1	25.33	88.28	89.08	89.30	**89.60**	69.06	76.03	20.14	20.07	25.00	20.69
news-5cl-2	30.11	82.38	84.16	83.66	83.88	64.66	58.03	19.17	20.75	23.85	20.55
news-5cl-3	28.73	78.61	81.82	82.50	83.45	71.72	77.48	19.41	19.83	21.19	19.96

Table 4.4.: Fourth part of the classification accuracy in percent for the classical measures, obtained on each dataset and by taking the average of 5 experiments.

From the tables, we can make some observations.

- The measure that leads to the best outcome depends on the dataset. Thus, we can not directly ascertain if one method is better than another; it is often useful to test them to find the best one.
- However it is irrelevant to compare the average result over all the datasets without standardizing them, a brief look allows to obtain simple indications. We discover that the most ideal method is the *regularized commute-time kernel* which generates a mean of 83.19%. Nevertheless, this approach yields the top score only for the *IMDB dataset*. A more precise look at the findings reveals that this is the dataset for which the other methods often obtain their worse results, with only 53.83% of labels found. We can therefore confirm that *regularized commute-time kernel* has the best average outcome only thanks to its top score on this dataset, and the mean is not a sufficient statistical tool to find the most suitable technique.
- Counting the number of high scores obtained by the techniques generates the *corrected commute-time kernel*, which leads with five top results. Nonetheless, this produces only the sixth average outcome. It is thus also insufficient to determine the best method neither and more advanced statistical tools are needed to gain insight into the performance of similarity and dissimilarity measure.

4.2.1. Illustrations of some results

After viewing the numbers displayed in tables, it may be helpful to have a visualization of a similarity matrix. This is accomplished by Figure 4.1, which illustrates the function for four different techniques. Of course, this type of representation can only be obtained if the dataset was originally organized by class. This is why I have selected the dataset *news-3cl-1*, whose the rows are not switched unlike in experiment.

The three first pictures in Figure 4.1 depict the kernel matrix for the free-energy distance, with multidimensional scaling and Gaussian mapping, and for the corrected commute-time kernel, respectively. These three techniques have results greater than 90%, and we can easily differentiate between the three classes (although 4.1c warrants a closer inspection due to the use of similar colors). Moreover, the difference between the two images at the top of the figure signalized the difference between the two transformations from distance into kernel. The changes, however, are minimal.

The fourth image indicates the matrix obtained by the Markov diffusion squared distance. It is clearly impossible to distinguish between the three classes in this case. Moreover, Table 4.2 suggests that the result is nearly 1/3 for three different classes. It may therefore be speculated that a node has been correctly labeled by chance.

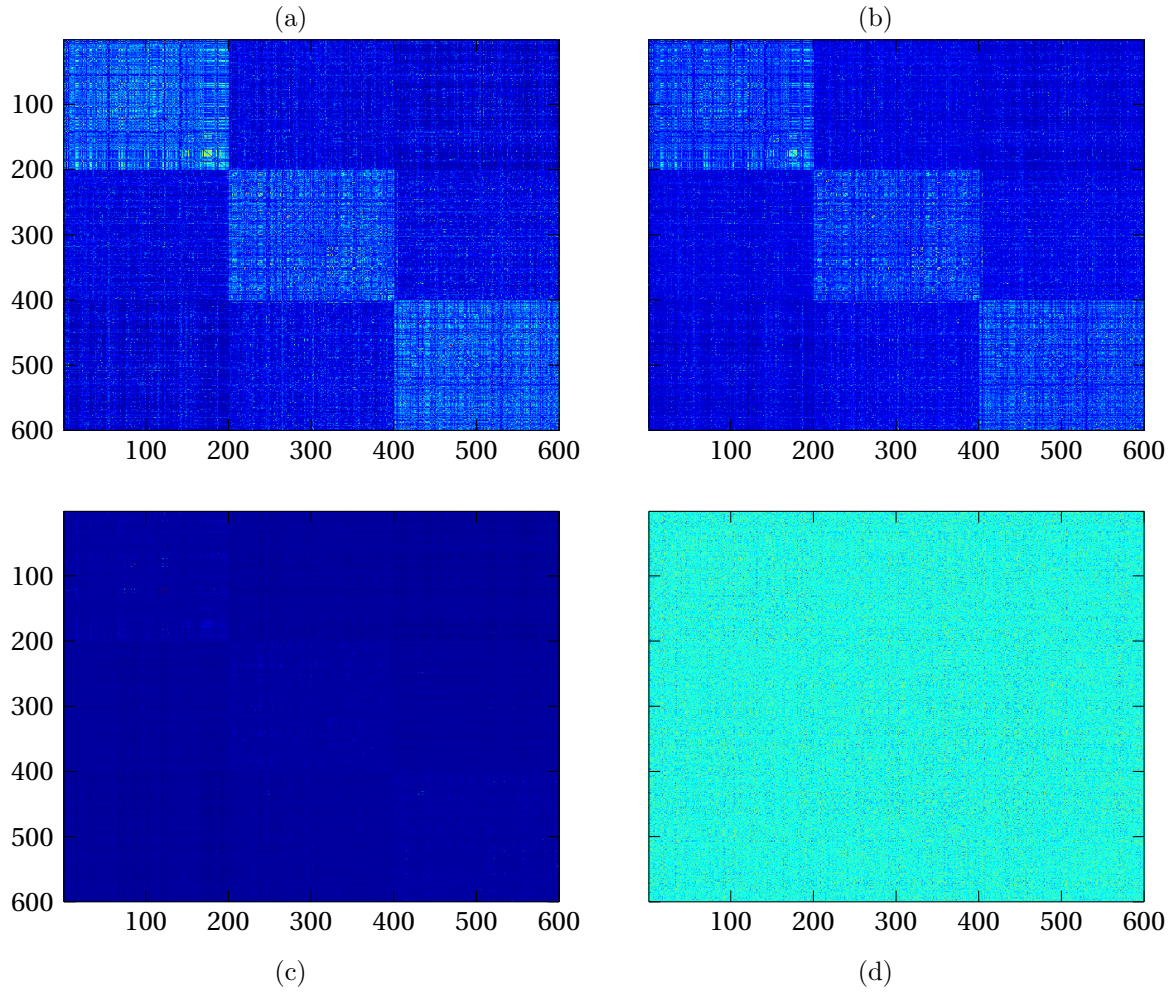

Figure 4.1.: Images of different similarity matrices for the dataset *news-3cl-1* obtained with the function `imagesc` of MATLAB™: (a) FE-mds, (b) Fe-g, (c) CCTK and (d) MDSD-mds.

4.2.2. Borda ranking

For an initial idea of the methods that produce the most favorable outcomes, I have performed a simple Borda ranking to evaluate the rank of each. The outcome of this comparison can be found in Table 4.5.

From this table, we observe that for the sum-of-similarities classification, multidimensional scaling generates better results overall than Gaussian mapping; however this is not true for every method, since there is a only a proportion two-thirds one-third in favor of multidimensional scaling. It is also apparent that, excepted the ones related to Markov chains (see section 2.4.8), the similarity measures that directly produce a kernel matrix (see sections 2.4.4, 2.4.5, 2.4.6 and 2.4.7) obtain excellent scores with 7 of the first 13 methods. We therefore have to analyze the differences between both the approaches and the transformations into kernel matrices.

Method	Rank	Score	Method	Rank	Score
FE-mds	1	578	CTD-g	24	304
CCTK	2	566	CCTD-g	25	299
RCTK	3	563	ECTD-g	26	291
FE-g	4	553	KS-mds	27	289
BoHPSD-mds	5	551	BoHPwZP-mds	28	286
LDK	6	549	TAMSD-g	29	281
LFD-mds	7	545	SR-g	30	275
RWWR	8	520	SR-mds	31	269
ECTD-mds	9	512	BoP-g	32	260
CTK	10	505	BoHPwZP-g	33	259
EDK	11	496	BoP-mds	34	248
BoHPSD-g	12	495	KS-g	35	245
MRLK	13	477	BoPAP-g	36	233
LFD-g	14	455	CECTD-mds	37	216
RSPD-g	15	410	TAMDK	38	213
CTD-mds	16	354	BoHP-g	39	202
BoPC-mds	17	343	MDSD-g	40	201
BoPC-g	18	333	TAMSD-mds	41	190
LE-mds	19	329	CECTD-g	42	185
RSPD-mds	20	327	CCTD-mds	43	175
BoHP-mds	21	325	MDK	44	126
LE-g	22	308	MDSD-mds	45	96
BoPAP-mds	23	306			

Table 4.5.: Ranking of the different classical measures according to Borda method.

Of the methods that lead to the 20 most ideal scores, only five are able to contain their two transformations in kernel. These are all considered as a family of dissimilarities, since they are an interpolation between the shortest-path and commute-time distances. Moreover, four of them - the free-energy distance (see Section 2.5.5), the bag-of-hitting-paths surprisal distance (see Section 2.5.4), the randomized shortest path dissimilarity (see Section 2.5.6) and the bag-of-paths covariance measure (see Section 2.5.8) - are derived from the bag-of paths framework. The last of these is the logarithmic forest distance (see Section 2.5.1).

Furthermore, the average results of the top 10 of Borda ranking reveal further details. If we omit the best mean outcome, which is detained by the regularized commute-time kernel due to its excellent performance on the IMDB dataset, we can observe that the maximum difference between two means is only 5.54%. This suggests that the best techniques have minor discrepancies between them. The finding is particularly interesting, as it highlights the divergence between multidimensional scaling of the free-energy distance and its Gaussian mapping - the latter is otherwise the only instance of Gaussian mapping among the first 10. Thus, it seems easier to choose automatically a transformation instead of a measure.

In summary, the Borda ranking provides initial insight into the methods that work well for the sum-of-similarities classification. For instance, it already appears that multidimensional

scaling is the preferable option for converting distance measures into kernel matrices. The similarity measures that produce kernels matrices directly also tend to yield satisfactory results. The final comment is that the differences between the best and subsequent scores are generally small, and thus, it is helpful to use several techniques.

4.2.3. Friedman-Nemenyi test

We now proceed to the Friedman test for ensuring that at least one of the methods is significantly different from the others. Although the number of techniques may render this unlikely, the test confirms this intuition. For the chi-square distribution, we obtain

$$\chi_F^2 = 0.0812 \times [28606 - 23805] = 389.6205. \tag{4.1}$$

This leads to the F-distribution:

$$F_F = \frac{13 \times \chi_F^2}{(14 \times 44) - \chi_F^2} = 22.374. \tag{4.2}$$

With $V_{crit} = F(44, 572) \approx 1.40$, the F-distribution is much larger than the critical value. We can therefore conclude that the measures are not all significantly equal and should be succeeded by a post-hoc Nemenyi test where the critical difference is

$$CD = 3.943 \sqrt{\frac{45 \times 46}{6 \times 14}} = 19.5737. \tag{4.3}$$

and the mean rank of each method is presented in Table 4.6[1].

Method	Mean	Method	Mean	Method	Mean	Method	Mean
KS-g	14.71	SR-g	15.36	TAMSD-mds	12.93	BoHPSD-mds	39.18
KS -mds	20.57	SR-mds	18.36	MDK	8.79	FE-g	38.04
LE-g	17.07	EDK	35.36	TAMDK	14.86	FE-mds	41.29
LE-mds	19.32	LDK	38.57	LFD-g	30.32	RSPD-g	29.21
CTD-g	19.89	MRLK	34.00	LFD-mds	38.86	RSPD-mds	23.36
CTD-mds	25.14	CTK	35.46	BoP-g	15.57	BoPAP-g	13.75
ECTD-g	17.36	CCTK	40.07	BoP-mds	14.00	BoPAP-mds	17.79
ECTD-mds	35.96	RCTK	40.00	BoHPwZP-g	15.82	BoPC-g	21.39
CCTD-g	16.29	RWWR	37.11	BoHPwZP-mds	15.89	BoPC-mds	22.96
CCTD-mds	10.89	MDSD-g	12.79	BoHP-g	12.39		
CECTD-g	10.93	MDSD-mds	6.64	BoHP-mds	23.21		
CECTD-mds	12.89	TAMSD-g	17.96	BoHPSD-g	32.68		

Table 4.6.: Mean rank of the different methods.

Figure 4.2 offers a graphical representation of the follow-up evaluation. An inspection of the results indicates that 26 of the methods are significantly worse than the best option: the

[1] It is useful to notice in the MATLAB™ function used here, the rank are given in reverse order, i.e., rank 1 for the worst method, and so on... This changes nothing in the graphical representation.

free-energy distance with multidimensional scaling transformation. Due to the number of approaches tested, we have to tighten the criterion. Two methods are declared significantly different if the discrepancy between their means is larger than a half of the critical value.

By doing so, only 13 techniques remain in the top category - precisely the first 13 methods in the Borda ranking. We therefore surmise that they seem much better than the others. However, unless the first three methods retain their positions, we can observe some changes in the others. This is why more statistical tests are needed, but we focus henceforth on the 13 approaches mentioned above.

4.2.4. Wilcoxon signed-rank test

The final assessment of the most favorable classical measures is a Wilcoxon signed-rank test, whose results are displayed in Table 4.7. The legend for this representation is as follows:

- ▶ √ means that the method for the row is significantly better than that of the column.
- ▶ × represents the reverse of the above scenario.
- ▶ = denotes than either of the two methods is significantly better than the other.
- ▶ ∗ is a special case between the Euclidean commute-time distance with multidimensional scaling transformation and the commute-time kernel. Indeed, since the two methods have identical results for all *news datasets*, N_r is too small to find a critical value (see table 3.3) and thus the test does not yield an answer.

Method ↓	Number ↓ →	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.
FE-mds	1.	\	=	√	√	√	√	√	√	√	√	√	√	√
CCTK	2.	=	\	=	√	√	√	√	√	√	√	√	√	√
RCTK	3.	×	=	\	=	×	√	=	√	√	√	√	√	√
BoHPSD-mds	4.	×	×	=	\	√	=	=	√	√	√	√	√	√
LFD-mds	5.	×	×	√	×	\	√	=	√	√	√	√	√	√
LDK	6.	×	×	×	=	×	\	=	√	√	√	√	√	√
FE-g	7.	×	×	=	=	=	=	\	=	√	√	=	√	√
RWWR	8.	×	×	×	×	×	×	=	\	=	=	√	√	√
ECTD-mds	9.	×	×	×	×	×	×	×	=	\	∗	√	√	√
CTK	10.	×	×	×	×	×	×	×	=	∗	\	√	√	√
EDK	11.	×	×	×	×	×	×	=	×	×	×	\	√	√
MRLK	12.	×	×	×	×	×	×	×	×	×	×	×	\	×
BoHPSD-g	13.	×	×	×	×	×	×	×	×	×	×	×	√	\

Table 4.7.: Wilcoxon signed-rank test for the best methods.

This table first suggests that the rankings of the previous tests are generally respected. Next, the Wilcoxon test follows the rfindings from the Nemenyi one; only two red crosses in the upper-right region of the table contradict it. We also note that the free-energy distance with multidimensional scaling transformation and the corrected commute-time kernel seem to perform the most ideally. Indeed, they are significantly better than the other methods, ex-

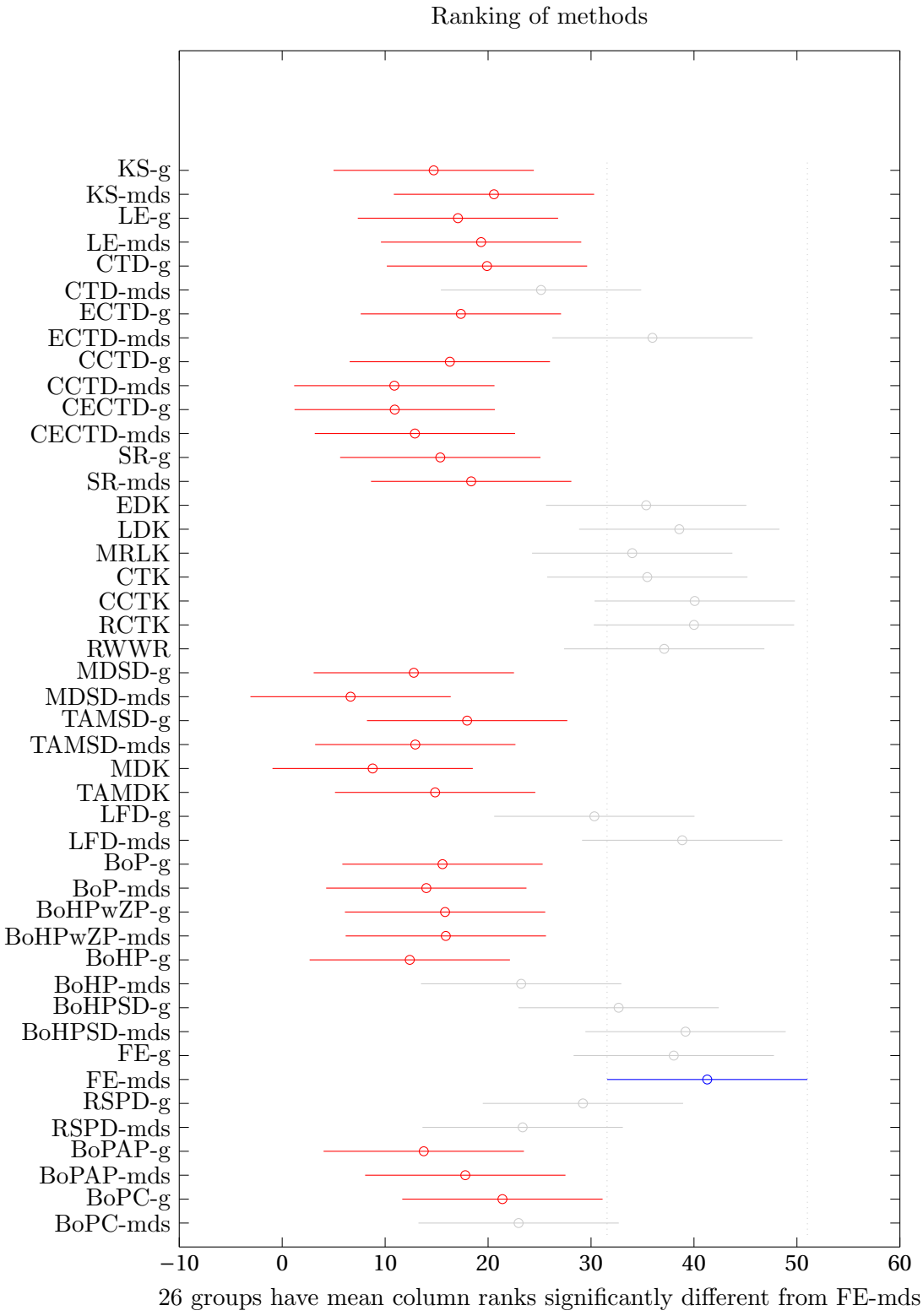

Figure 4.2.: Nemenyi representation for the post-hoc test.

cept the one directly following them - recall that these two represents 9 of the 14 high scores.

Furthermore, we can separate the 13 best methods into two groups. The first six methods are always significantly better than the last six, and they contain 13 out of 14 high scores. Only the surprisal distance with Gauss transformation yields another top score, but all the tests place it among the bottom two. The free-energy distance with the Gaussian mapping is an interesting case: despite its seventh position in the Nemenyi test, it fares only significantly worse than the first two and competes equally with the others four ranked above it. From the specific results, this seems to be because the free-energy methods are very accurate for the *news datasets* which represent 64% of the sources.

4.2.5. Comparison with the support vector machines classifier

Finally, we will compare our results with those obtained with another semi-supervised classification test: the transductive and spectral support vector machines (SVM). Details about this assessment and its results can be found in [32]. As expected, we have only included methods evaluated with both of classification tests - i.e., the free-energy distance; the bag-of-hitting-paths surprisal distance; the logarithmic forest distance; the randomized shortest-path dissimilarities; as well as the regularized commute-time kernel; the modified regularized Laplacian kernel and the Markov diffusion kernel.

The Borda rankings[2], reveal that the free-energy distance is always on top but with a different transformation. In both cases, the second conversion is ranked third, which indicates that this method is excellent. In general, the sum-of-similarities have better results with multidimensional scaling whereas support vector machines seems to prefer Gaussian mapping.

Another intriguing observation is that some methods provide ideal outcomes with one test but unsatisfactory ones with another. For instance, the regularized commute-time kernel is deemed one of the top performers according to the sum-of similarities, but worse than the Markov diffusion kernel based on the SVM. In fact, this kernel has nearly the worst results with from our experiment. In general, with the exception of the Markov diffusion case, the sum-of-similarities classification is favorable to measures that form a kernel matrix directly, while the SVM ranks these at bottom.

The final question is whether one classification test is better than another. To answer this, we examine the exact results of both. From the 154 percentages compared, the SVM yields a better outcome in 83 cases, while the sum-of-similarities has only 70. The last case is an equality. On the other hand, the sum-of-similarities leads to a greater number of top results

[2]The borda rankings are of course modified to only leave the techniques cited above. The scores are thus now irrelevant.

for 6 of the 11 methods compared. Sometimes, one approach consistently generates the best outcomes for almost all datasets, and the evaluation is clear, but with other measures, both are comparable in quality. In the latter case, it is much more difficult to choose a winner, and it appears that none of the tests is significantly better than any other.

4.3. Results of the randomized optimal transport on a graph

We now explore the results obtained by the original method introduced in Section 2.6. As previously discussed, this distance can be computed in two ways, depending on whether we consider the hitting-paths or not. With two transformations into a kernel matrix, this leads to four new techniques for analysis. The outcomes for all the datasets are depicted in Table 4.8. The ones that are higher than the best result from the classical measures are in boldface (see Tables 4.1, 4.2, 4.3 and 4.4). The distributions for the "supply" and "demand" nodes are uniform - i.e., $\sigma^i_{\text{in}} = 1/n$ and $\sigma^i_{\text{out}} = 1/n$ for all $i = 1, \ldots, n$. There are probably distributions which would get better results but uniform distributions have been chosen so as not to favor certain nodes as it is the case in the other methods.

For the assessment of these new methods, we retain only the most favorable classical ones, namely those that are close enough to the free-energy with multidimensional scaling.

Measure → Dataset ↓	ROTP wHP-g	ROTPw HP-mds	ROT -g	ROT -mds
texas	48.80	79.49	48.80	**80.24**
washington	67.28	**69.30**	60.03	68.03
wisconsin	44.68	75.21	45.26	**75.50**
cornell	41.91	58.82	41.91	59.90
imdb	50.98	50.02	50.98	50.17
news-2cl-1	**95.88**	95.25	94.63	95.00
news-2cl-2	92.72	92.90	91.21	91.14
news-2cl-3	96.11	96.18	95.99	95.99
news-3cl-1	93.79	93.50	92.96	92.92
news-3cl-2	92.22	93.35	89.72	92.35
news-3cl-3	90.04	92.27	87.44	91.30
news-5cl-1	88.53	**89.63**	87.00	89.15
news-5cl-2	83.53	83.98	81.23	82.23
news-5cl-3	79.66	83.58	77.26	81.04

Table 4.8.: Results of the classification accuracy in percent for the randomized optimal transport, obtained on each dataset and by taking the average of 5 experiments.

In this table, the results of the two versions seem to be in the order of the best classical measures. A clue for this observation is the five top scores obtained using the randomized optimal transport (three with the hitting-paths and two without). While all the variations yield satisfactory results, the one in the hitting-paths case appears preferable to the one

without. Again, the multidimensional scaling transformation has produced a better outcome than the Gaussian mapping in most instances.

4.3.1. Illustrations of some results

After observing the results, it may be helpful to have a visual interpretation of the kernel matrices. The representation of the four variations can be found in Figure 4.3; they are again performed with the *news-3cl-1* dataset.

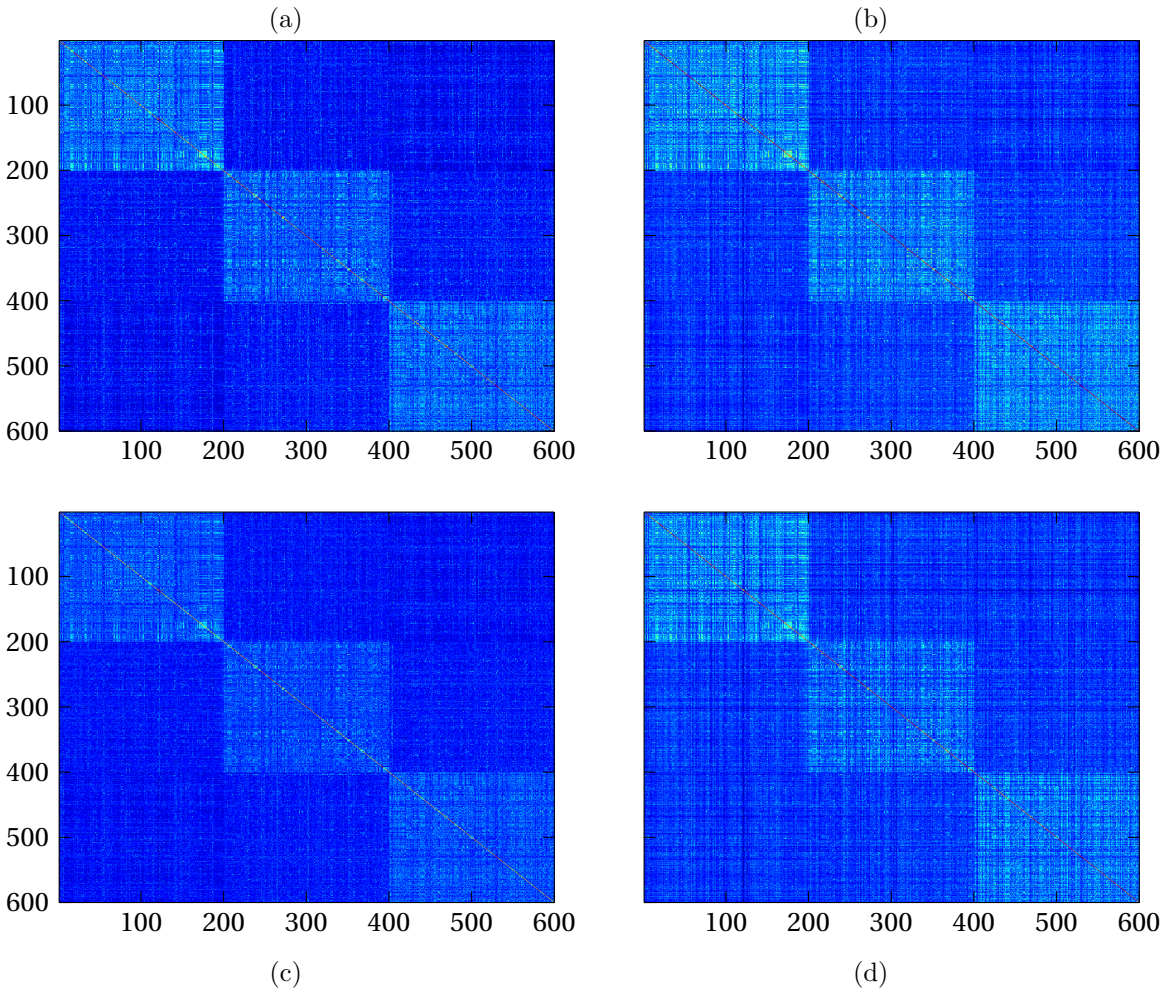

Figure 4.3.: Images of different similarity matrices for the dataset *news-3cl-1* obtained with the function `imagesc` of MATLAB™: (a) ROTwHP-mds, (b) ROTwHP-g, (c) ROT-mds and (d) ROT-g.

As predicted with the results over **90%** from Table 4.8, the three classes are clearly differentiable. Moreover, by examining the digital outcomes, we postulate that the hitting-paths case is better than its alternative; this notion is reinforced by the diagrams. Indeed, the distinctions between the classes are more pronounced in the top images- which correspond to randomized optimal transport with hitting-paths - than the ones below them.

4.3.2. Borda ranking

To confirm our intuitions, we will turn to the Borda ranking that can be found in Table 4.9.

Method	Rank	Score
ROTwHP-mds	1	203
FE-mds	2	202
CCTK	3	181
BoHPSD-mds	4	167
FE-g	5	159
ROT-mds	6	154
LFD-mds	7	151
RCTK	8	143
ROTwHP-g	9	138
LDK	10	127
ECTD-mds	11	111
CTK	12	110
BoHPSD-g	13	100
RWWR	14	92
ROT-g	15	74
EDK	16	65
MRLK	17	55

Table 4.9.: Ranking of the best classical and the randomized optimal transport methods according to Borda method.

Several observations can be made from this table. First, even if the classical methods are not in the same order as before, the trends remain identical and it is not necessary to comment on the minor changes. We can thus focus solely on the manifestations of the techniques within the randomized optimal transport framework.

Next, of the positions obtained through the randomized optimal transport, three appear among the 10 best methods. Better still, the randomized optimal transport with the hitting-paths and multidimensional scaling is first in the Borda ranking even if the best free-energy distance is very close behind. Conversely, the fourth technique with Gaussian mapping but without the hitting-paths is ranked in the bottom and seems unable to compete with the top methods.

A final observation is that the two variations with multidimensional scaling fare better than their Gaussian counterparts. We may therefore suppose that for the randomized optimal transport scenario, the choice of transformation into a kernel matrix has a greater impact than the presence of hitting-paths.

4.3.3. Friedman-Nemenyi test

We now perform the Friedman test again to determine if, with fewer methods, at least one is significantly worse than another. This time, we obtain for the chi-square function

$$\chi_F^2 = 0.549\,[1532.018 - 1377] = 85.108. \tag{4.4}$$

The resulting F-distribution is therefore

$$F_F = \frac{13 \times 85.108}{14 \times 16 - 85.108} = 7.996. \tag{4.5}$$

We have to compare this figure with the critical value $V_{crit} = F(16, 208) \approx 1.69$. Again, the computed value is larger than the critical one, and thus significant differences exist between the measures. We therefore proceed with a post-hoc Nemenyi test. The mean ranks can be found in Table 4.10 and the critical difference calculated as

$$CD = 3.458\sqrt{\frac{17 \times 18}{6 \times 14}} = 6.60. \tag{4.6}$$

Method	Mean	Method	Mean
ECTD-mds	7.2857	BoHPSD-g	6.5714
EDK	4.5714	BoHPSD-mds	11.6071
LDK	8.3214	FE-g	10.8214
MRLK	4.5714	FE-mds	14.1071
CTK	7.2143	ROTwHP-g	9.2143
CCTK	12.5000	ROTwHP-mds	13.9286
RCTK	9.8929	ROT-g	4.7857
RWWR	6.3929	ROT-mds	10.6429
LFD-mds	10.5714		

Table 4.10.: Mean rank of the different methods.

Figure 4.4 depicts the Nemenyi evaluation graphically. It illustrates several things. First, the top method here is not the same as in the Borda ranking. Indeed, even if the difference is small, the free-energy distance is more accurate than all variations of randomized optimal transport.

Next, the best technique is significantly better than seven other methods, including one from this new framework - the version with Gaussian mapping and no hitting-paths. Our predictions may therefore seem to be correct, and this can be definitely excluded from our methods of interest. Ten methods remain, among which the Nemenyi test does not differentiate significantly and for which pairwise comparisons may be useful. Many have already been conducted in Section 4.2.4, the next section focus on the comparisons between the remaining three variants of randomized optimal transport, and all 10 methods that have passed this last test.

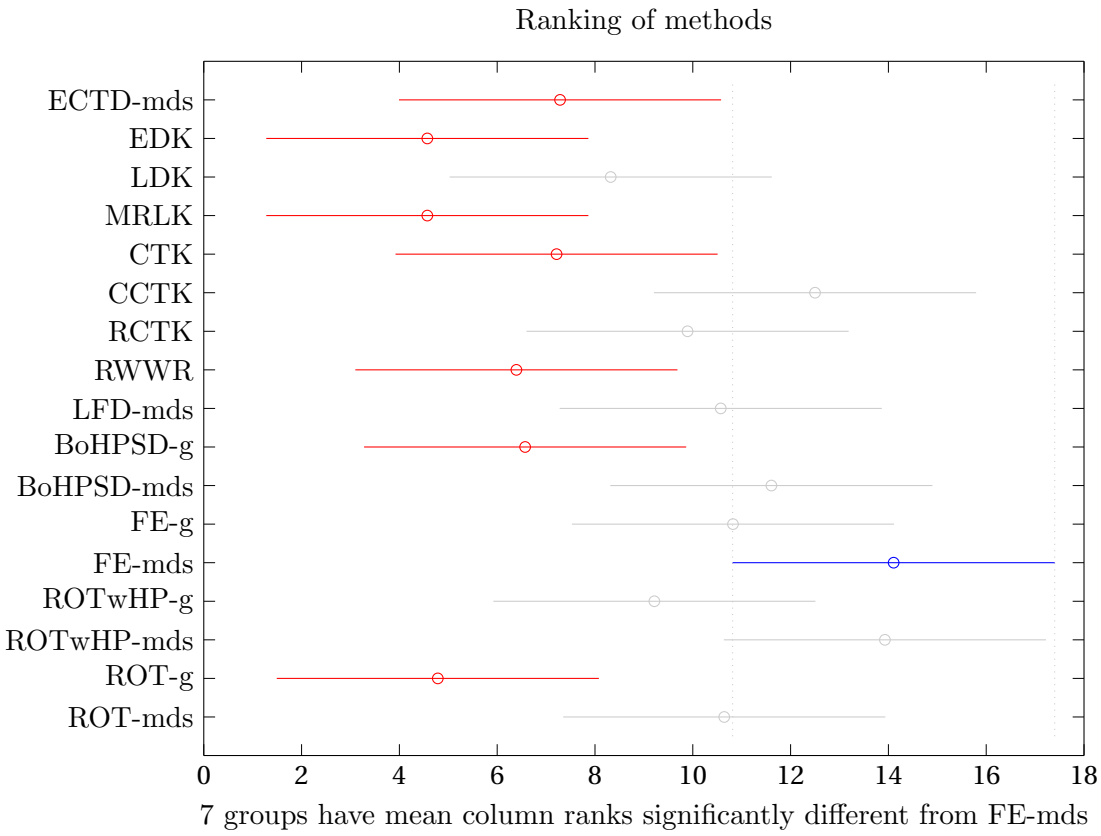

Figure 4.4.: Nemenyi representation for the post-hoc test.

4.3.4. Wilcoxon signed-rank test

I have subsequently made pairwise evaluations with a Wilcoxon signed-rank test. The results can be found in Table 4.11 in which the numbers correspond to:

1. FE-mds
2. ROTwHP-mds
3. CCTK
4. BoHPSD-mds
5. FE-g
6. ROT-mds
7. LFD-mds
8. RCTK
9. ROTwHP-g
10. LDK

First, we note that the free-energy distance with multidimensional scaling is significantly more accurate than all those using randomized optimal transport; thus, it seems to be the most reliable method.

Nevertheless, the first technique in this framework is still demonstrably better than all the others.

For the comparisons implying the ROT-mds, we notice that, except for the two best methods, it performs at least as well as the others. It therefore seems to be an adequate alternative

Method	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.
ROTwHP-mds	×	\	√	√	√	√	√	√	√	√
ROT-mds	×	×	=	=	=	\	=	√	√	√
ROTwHP-g	×	×	×	×	×	×	×	×	\	=

Table 4.11.: Wilcoxon signed-rank test for the best randomized optimal transport methods.

and may be used in some situations. On the other hand, the two version with Gaussian mapping are significantly worse than the top choices; our previously stated prediction is confirmed, and the option of multidimensional scaling is preponderant compared to the presence of hitting-paths.

4.4. Conclusion

We can summarize the above discussions and make some relevant assumptions in selecting the most appropriate method for the respective case.

From the Borda ranking and Friedman-Nemenyi test, we have observed that, of the classical measures introduced in [29], 13 demonstrated better performances than the others. Through the digital results, these methods reveal only minor differences among themselves, and this is difficult to separate them with the present information.

To this end, some pairwise comparisons have been made; they suggest that six methods are nearly significantly better than all the others. These methods have yielded 13 high scores on the 14 datasets used for the tests. Moreover, if we use a minimum number of approach for the sake of simplicity, we would note that two of them - free-energy distance with multidimensional scaling and corrected commute-time kernel - can be classified separately. Although this top category would only includes two techniques, 11 of the 14 high scores would still be in it. We may hereby conclude than perhaps these two methods would almost always generate the best results.

However, [38] has recently introduced a new framework - randomized optimal transport -leading to four new variants that can compute similarities between nodes. The measures introduced in this context have overtaken the top-performing classical methods for five datasets. This is only the case when the high score was originally detained by a family of dissimilarities. It was therefore interesting to compare the recent techniques with the best traditional measures.

All the statistical tests have demonstrated that most of these new approaches can compete against classical ones. Furthermore, the best of them - randomized optimal transport with hitting-paths and multidimensional scaling transformation - is very close to the free-energy

distance and can be grouped with the two best traditional measures.

A final comment can be made about the kernel matrix transformations used here. All the experiments suggest that multidimensional scaling outperforms Gaussian mapping. The latter may be rejected, but only for the case of the sum-of-similarities classification test used in this thesis. Indeed, the SVM variant utilized in [32] leads to the opposite observation and produces better results for Gaussian mapping. We are therefore unable to choose a default transformation and both of have to be considered. Speaking of SVM, it is interesting to note this classifier sometimes obtain worse results than sum-of-similarities. Since SVM is more sophisticated and requires much more computational resources, it could have been expected that it would always be more accurate.

In summary, since the results depend on the dataset, and one method is not always better than the others, it is impossible to select a definitive measure that would yield the best outcome in every situation. Nevertheless, we can identify a subset of the methods that generate the optimal results in most cases.

After these conclusions, we will now talk about the limitations we have been subjected to in this master thesis. This will be done in the next chapter where we will also give some ideas to improve our observations.

Limitations and further research

5

Contents

5.1. Preliminaries

As with all research topic, the one treated in this thesis is subject to limitations. Thus, this chapter presents a non-exhaustive list thereof that has been encountered during the realization of this work. It also offers some ideas to overcome these constraints in order to specify the results and improve the conclusions. Following this, I review directions for further studies that may be of interest.

5.2. Research limitations

This subchapter addresses the limitations and ideas to improve them. Of course, the realization of these suggestions may lead to new research, but unlike the recommendations introduced in the next section, these would merely require resuming the above investigations with another set of parameters.

First, since the data represent real-world situations, their quantity is limited. Fourteen datasets are insufficient to draw definitive conclusions. More information would have been preferable to refine the trends observed in this thesis. Moreover, the material was taken from only three databases, and the methods have often yielded similar results for each. We therefore have no idea how measures would fare if they were tested on graphs depicting completely different subjects, for which the adjacency matrix may have various structures. The computational capacity also restricted the study to small graph, with the largest dataset comprising slightly over 1000 nodes. It would be interesting to observe the performance of the methods with a larger group of data. Furthermore, there is a maximum of 6 categories in which the vertices are labeled, and it would have been insightful to investigate relationships between the outcomes, the size of the graphs and the numbers of classes.

Second, most of the dissimilarity measures depend on parameters. Again, due to computational restrictions, I was only able to test finite range of figures to this end. Using more parameter values would have made the findings more precise. To confirm the quality of a method, it would be helpful to optimize such values over a continuous domain, but this would require a considerable amount of work in the optimization area.

Third, I have only computed the percentages of labels for training sets of 20% of the total number of nodes. It would have been interesting to explore the effects when this proportion changes (for example, if it becomes really small) and whether there is a link between the size of the training sets and the results obtained.

Fourth, to select the best method, I have restricted the investigation to only three statistical tests. To clarify the present conclusions or uncover new ones, we can use additional tools. Instead of contrasting the techniques, it would be intriguing to compare the findings to a baseline and determine a performance scale.

Last, the performance index is based only on the percentage of found nodes. We could vary this figure to quantify the quality of a measure. For instance, it may be useful to analyzed the computation time and ask these questions :

- What is the time complexity of the methodology ? How does the computation time increase when a greater number of nodes is used ?
- Up to what point is an improvement in accuracy meaningful if it doubles or triples the computation time ?

5.3. Improvements and ideas for further works

This segment considers some directions for future development in the subject area. These propositions are wide-ranging and may form the basis for further research articles and even master's theses.

In this investigation, the methods have only been tested with one classification algorithm, the sum-of-similarities. I have compared the results with another assessment technique introduced in [32] - the transductive and spectral support vector machines. It may be worthwhile to evaluate the various measures with other tools such as an alternative semi-supervised classification task (see, e.g., [90, 103]), an unsupervised algorithm or parts of the clustering framework.

I have also noted that the transformation of a distance measure into a kernel matrix can be done in two ways. However, these conversions lead to inconsistent outcomes depending on the approach or the classification test. It may be interesting to explore newer and more

reliable means of performing this transformation.

In addition, a recently introduced framework (see [38]) has been demonstrated to be comparable with the classical methods. This example illustrates the value in analyzing new approaches to improve the precision of existing tests.

Finally, I have indicated that some methods only function for graphs with certain characteristics, such as directed or weighted networks. It may be maybe worthwhile to consider improvements using these properties and the resulting adjacency matrix structure. For instance, the sparse nature of the matrix may reduce the computation time.

Conclusion

6

This thesis has mainly tried to answer the question : how can we quantify the degree of connection between nodes ? To this end, I have presented various measures between the vertices of a graph. These may be similarities, dissimilarities or distances, and in many cases, only the adjacency matrix of the network needs to be computed.

First, I discussed similarity measures, most of which are mere modifications or improvements of the two basic intervals: the shortest-path and the commute-time distances. Some of them have provided accurate results with the experimental procedures, especially when the measure yields a kernel matrix directly, such that no conversion is necessary.

Conversely, I have also presented a series of measures which known as families of dissimilarities, because they interpolate between the shortest-path and the commute-time distance. This interpolation depends on a parameter, from whose extreme values we can retrieve the basic distance. Most of these families come from a unique framework, the bag-of-paths probabilities, which leads to many techniques that generate excellent results. Among them, the free-energy distance seems to be the one with the most consistent outcomes when transformed into a kernel matrix with multidimensional scaling.

Although some of these measures provide satisfactory conclusions, I have decided to address one more method. This was recently introduced in [38] and I wanted to compare it with the other top approaches; it performed even better on some datasets. Moreover, one of its variants seems to be almost equally effective, according to the statistical tests used in this thesis. While many methods have already been developed, it is always useful to search for new techniques, with diverse properties that enable them to perform on a wide range of data.

Finally, we have made the assumption that it is impossible to choose one default approach among the ones that generate the best results. Indeed, the techniques are datasets-dependent, and the possibility of a poor outcome cannot be eliminated even for the apparent top measure. We have therefore attempted to constitute a set of methods in which the likelihood of obtaining the high score is as great as possible. It is expected that in most of cases, at least one of these would satisfy the user. Nevertheless, due to the small number and size of sources tested, we cannot predict how various measures would react to new datasets; this is especially true in scenarios with much larger than those used in this thesis.

In conclusion, even if this study provides insight into the efficiency of a wide range of distance measures, all the observations made here have to be evaluated with more datasets

and classification tasks before we can make definitive inferences. As such, further research should be conducted in hopes of identifying the best method in the near future.

Bibliography

[1] Charu C Aggarwal and Chandan K Reddy. *Data clustering: algorithms and applications*. CRC press, 2013. (Cited on page 6.)

[2] Ravidra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. Network flows : theroy, algorithms and applications. Prentice hall, Upper Saddle River, New Jersey, Etats-Unis, 1993. (Cited on page 20.)

[3] Morteza Alamgir and Ulrike von Luxburg. Phase transition in the family of p-resistances. In *Proceedings of the 24th International Conference on Neural Information Processing Systems*, NIPS'11, pages 379–387, USA, 2011. Curran Associates Inc. (Cited on page 14.)

[4] A. Albert. *Regression and the Moore-Penrose Pseudoinverse*. Mathematics in Science and Engineering. Elsevier Science, 1972. (Cited on page 7.)

[5] H. Anton and R.C. Busby. *Contemporary Linear Algebra*. Wiley, 2003. (Cited on page 7.)

[6] A. L. Barabasi. *Network science*. To appear at Cambridge University Press; preprint available from http://barabasi.com/networksciencebook, 2016. (Cited on page 1.)

[7] S. Barnett. *Matrices: Methods and applications*. Oxford University Press, 1992. (Cited on page 7.)

[8] A. Ben-Israel and T. Greville. *Generalized inverses: Theory and applications*. Springer, 2nd ed., 2003. (Cited on page 7.)

[9] Bela Bollobas. *Modern graph theory*. Springer, 2nd edition, 1998. (Cited on page 4.)

[10] Ingwer Borg and Patrick Groenen. Modern multidimensional scaling: theory and applications. 1997. (Cited on page 9.)

[11] G. Brightwell and P. Winkler. Maximum hitting time for random walks on graphs. *Random structures and algorithms*, 1:263–276, 1990. (Cited on page 10.)

[12] S. Brin and L Page. The anatomy of a large-scale hypertextual web search engine. 30(1-7):107117, 1998. (Cited on page 13.)

[13] P. Chebotarev and E. Shamis. The matrix-forest theorem and measuring relations in small social groups. *Automation and Remote Control*, 58:15051514, 1997. (Cited on pages 12 and 14.)

[14] P. Chebotarev and E. Shamis. On proximity measures for graph vertices. *Automation and Remote Control*, 59:14431459, 1998. (Cited on page 12.)

[15] Pavel Chebotarev. A class of graph-geodetic distances generalizing the shortest-path and the resistance distances. *Discrete Applied Mathematics*, 159(5):295 – 302, 2011. (Cited on page 14.)

[16] Pavel Chebotarev. The graph bottleneck identity. *Advances in Applied Mathematics*, 47(3):403 – 413, 2011. (Cited on page 14.)

[17] Pavel Chebotarev. The walk distances in graphs. *Discrete Applied Mathematics*, 160(10):1484 – 1500, 2012. (Cited on page 14.)

[18] Fan RK Chung. *Spectral graph theory*. Number 92. American Mathematical Soc., 1997. (Cited on page 6.)

[19] R. R. Coifman, S. Lafon, A. B. Lee, M. Maggioni, B. Nadler, F. Warner, and S. W. Zucker. Geometric diffusions as a tool for harmonic analysis and structure definition of data: Diffusion maps. *Proceedings of the National Academy of Sciences of the United States of America*, 102(21):7426–7431, 2005. (Cited on page 13.)

[20] T. Cox and M. Cox. *Multidimensional scaling, 2nd ed.* Chapman and Hall, 2001. (Cited on page 9.)

[21] J.-C. de Borda. Mémoire sur les élections au scrutin. *Mémoire de l'académie royale des sciences*, pages 657–664, 1781. (Cited on page 32.)

[22] J. Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006. (Cited on pages 33 and 34.)

[23] Robin Devooght, Amin Mantrach, Ilkka Kivimäki, Hugues Bersini, Alejandro Jaimes, and Marco Saerens. Random walks based modularity: application to semi-supervised learning. In *Proceedings of the 23rd international conference on World wide web (WWW '14)*, pages 213–224. ACM, 2014. (Cited on page 15.)

[24] M. Deza and E. Deza. *Encyclopedia of distances*. Springer, 3rd ed., 2014. (Cited on pages 7 and 8.)

[25] P.G. Doyle and J.L. Snell. *Random walks and electric network*. The Mathematical Association of America, 1984. (Cited on page 18.)

[26] Reichl L. E. *A modern course in statistical physics*. Wiley, 2nd ed., 1998. (Cited on page 19.)

[27] Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, Sep 1936. (Cited on page 8.)

[28] Ernesto Estrada. *The structure of complex networks*. Oxford University Press, 2012. (Cited on page 1.)

[29] F Fouss, M Saerens, and Shimbo. *Algorithms and Models for Network Data and Link Analysis*. Cambridge University Press, 2016. (Cited on pages 2, 4, 7, 9, 10, 11, 12, 13, 14, 16, 17, 18, 19, 20, 39 and 54.)

[30] Francois Fouss, Alain Pirotte, Jean-Michel Renders, and Marco Saerens. Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 19(3):355–369, March 2007. (Cited on page 12.)

[31] Francois Fouss, Luh Yen, Alain Pirotte, and Marco Saerens. An experimental investigation of graph kernels on a collaborative recommendation task. In *Proceedings of the Sixth International Conference on Data Mining*, ICDM '06, pages 863–868, Washington, DC, USA, 2006. IEEE Computer Society. (Cited on page 14.)

[32] Kevin Françoisse, Ilkka Kivimäki, Amin Mantrach, Fabrice Rossi, and Marco Saerens. A bag-of-paths framework for network data analysis. *Neural Networks*, 90:90 – 111, 2017. (Cited on pages 2, 14, 15, 16, 17, 21, 23, 24, 35, 48, 55 and 58.)

[33] M. Friedman. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, 32:675–701, 1937. (Cited on page 33.)

[34] M. Friedman. A comparison of alternative tests of significance for the problem of m rankings. *Annals of Mathematical Statistics*, 11:86–92, 1940. (Cited on page 33.)

[35] M. Gondran and M. Minoux. *Graphes et algorithmes*. Collection EDF RD, 2009. (Cited on page 4.)

[36] J. C. Gower. Similarity, dissimilarity and distance, measures. In S. Kotz, N. Balakrishnana, C. Read, B. Vidakovic, and N. Johnson, editors, *Encyclopedia of statistical science*. Wiley, 2004. (Cited on pages 7 and 8.)

[37] Charles Miller Grinstead and James Laurie Snell. *Introduction to probability*. The Mathematical Association of America, 2nd ed., 1997. (Cited on pages 6 and 18.)

[38] G. Guex, I. Kivimäki, and M. Saerens. Randomized optimal transport on a graph: Framework and applications. (Cited on pages 2, 20, 21, 22, 54, 59 and 61.)

[39] R.L. Iman and J. M. Davenport. Approximations of the critical region of the friedman statistic. *Communications in Statistics*, pages 571–595, 1980. (Cited on page 33.)

[40] T. Ito, M. Shimbo, T. Kudo, and Y. Matsumoto. Application of kernels to link analysis. In *the 11th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 05)*, pages 586–592, 2005. (Cited on page 12.)

[41] P. Jaccard. Étude comparative de la distribution florale dans une portion des alpes et du jura. *Bulletin de la Société Vaudoise des Sciences Naturelles*, 37:547 – 579, 1901. (Cited on page 8.)

[42] M. James. The generalised inverse. *The Mathematical Gazette*, 62(420):109114, 1978. (Cited on page 7.)

[43] N. Jardine. *Mathematical taxonomy*. Wiley, 1971. (Cited on pages 7 and 8.)

[44] E. T. Jaynes. Information theory and statistical mechanics. *Physical Review*, 106:620–630, May 1957. (Cited on page 19.)

[45] Glen Jeh and Jennifer Widom. Simrank: a measure of structural-context similarity. In *Proceedings of the e8th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '02)*, pages 538–543, New York, NY, USA, 2002. ACM Press. (Cited on page 11.)

[46] Leonid Vitaliyevich Kantorovich. On the translocation of masses. In *In Dokl. Akad. Nauk SSSR*, volume 37, pages 199–201. MAIK Nauka/Interperiodica, 1942. (Cited on page 20.)

[47] M. Kardar. *Statistical physics of particles*. Cambridge University Press, 2007. (Cited on page 19.)

[48] Leo Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18(1):39–43, Mar 1953. (Cited on page 10.)

[49] John G Kemeny and James Laurie Snell. *Finite markov chains*. Springer, 1976. (Cited on pages 6, 10, 16 and 18.)

[50] I. Kivimäki, M. Shimbo, and M. Saerens. Developments in the theory of randomized shortest paths with a comparison of graph node distances. *Physica A: Statistical Mechanics and its Applications*, 393:600–616. (Cited on pages 14, 15, 17 and 18.)

[51] Eric D. Kolaczyk. *Statistical analysis of network data: Methods and models*. Springer Series in Statistics. Springer, 2009. (Cited on pages 1 and 4.)

[52] Risi Imre Kondor and John Lafferty. Diffusion kernels on graphs and other discrete input spaces. In *in: Proceedings of the 19th International Conference on Machine Learning (ICML '02)*, pages 315–322. Morgan Kaufmann, 2002. (Cited on page 11.)

[53] S. Kung. *Kernel methods and machine learning*. Cambridge University Press., 2014. (Cited on page 8.)

[54] Amy N. Langville and Carl D. Meyer. A survey of eigenvector methods for web information retrieval. *SIAM Review*, 47(1):135–161, 2005. (Cited on page 13.)

[55] B. Lebichot, I. Kivimaki, and M. Saerens. Bag-of-paths absorption probabilities for semi-supervised classification on graphs. *Submitted for publication*, 2015. (Cited on pages 18 and 19.)

[56] B. Lebichot, I. Kivimäki, K. Francoisse, and M. Saerens. Semi-supervised classification through the bag-of-paths group betweenness. *IEEE Transactions on Neural Networks and Learning Systems*, 25(6):1173–1186, 2014. (Cited on page 15.)

[57] P. Legendre and L. Legendre. *Numerical ecology*. Elsevier, 3rd ed., 2012. (Cited on pages 7 and 8.)

[58] E. A. Leicht, P. Holme, and M. E. J. Newman. Vertex similarity in networks. *Physical Review E*, 73:026120, Feb 2006. (Cited on page 10.)

[59] T. Lewis. *Network science*. Wiley, 2009. (Cited on page 1.)

[60] David Liben-Nowell and Jon Kleinberg. The link-prediction problem for social networks. *Journal of the American Society for Information Science and Technology*, 58(7):1019–1031, May 2007. (Cited on page 10.)

[61] D. Lin. An information-theoretic definition of similarity. In *Proceedings of the Fifteenth International Conference on Machine Learning*, ICML '98, pages 296–304, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc. (Cited on page 7.)

[62] Ulrike V. Luxburg, Agnes Radl, and Matthias Hein. Getting lost in space: Large sample analysis of the resistance distance. In J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 2622–2630. Curran Associates, Inc., 2010. (Cited on pages 1, 2, 10 and 12.)

[63] Ulrike V. Luxburg, Agnes Radl, and Matthias Hein. Hitting and commute times in large random neighborhood graphs. *Journal of Machine Learning Research*, 15:1751–1798, 2014. (Cited on pages 1 and 2.)

[64] Sofus A. Macskassy and Foster Provost. Classification in networked data: A toolkit and a univariate case study. *Journal of Machine Learning Research*, 8:935–983, 2007. (Cited on pages 35 and 36.)

[65] De Ron Malian. Template mémoire. http://www.asbo.com, 30/07/2016. (Cited on page 2.)

[66] Amin Mantrach, Nicolas van Zeebroeck, Pascal Francq, Masashi Shimbo, Hugues Bersini, and Marco Saerens. Semi-supervised classification and betweenness computation on large, sparse, directed graphs. *Pattern Recogn.*, 44(6):1212–1224, June 2011. (Cited on page 30.)

[67] Amin Mantrach, Luh Yen, Jerome Callut, Kevin Francoisse, Masashi Shimbo, and Marco Saerens. The sum-over-paths covariance kernel: A novel covariance measure between nodes of a directed graph. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(6):1112–1126, 2010. (Cited on pages 15, 19, 20, 21 and 23.)

[68] K.V. Mardia, J.T. Kent, and J.M. Bibby. *Multivariate analysis*. Academic Press, 1979. (Cited on pages 8 and 9.)

[69] Hairer Martin. Ergodic properties of markov processes. http://www.hairer.org/notes/Markov.pdf, March 9, 2006. (Cited on page 22.)

[70] B. Nadler, S. Lafon, R. Coifman, and I. Kevrekidis. Diffusion maps, spectral clustering and eigenfunctions of fokker-planck operators. In *In Advances in Neural Information Processing Systems 18: Proceedings of the NIPS 05 Conference*, pages 955–962. MIT Press, 2006. (Cited on page 13.)

[71] B. Nadler, S. Lafon, R. Coifman, and I. Kevrekidis. Diffusion maps, spectral clustering and reaction coordinate of dynamical systems. 21:113–127, 2006. (Cited on page 13.)

[72] P. B. Nemenyi. *Distribution-free multiple comparisons*. PhD thesis, Princeton University, 1963. (Cited on page 33.)

[73] M. E. J. Newman. *Networks: An introduction*. Oxford University Press, 2010. (Cited on pages 1 and 4.)

[74] James R Norris. *Markov chains*. Number 2. Cambridge university press, 1998. (Cited on pages 6 and 10.)

[75] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford University, November 1998. (Cited on page 13.)

[76] Jia-Yu Pan, Hyung-Jeong Yang, Christos Faloutsos, and Pinar Duygulu. Automatic multimedia cross-modal correlation discovery. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, pages 653–658, New York, NY, USA, 2004. ACM. (Cited on page 13.)

[77] L. Peliti. *Statistical mechanics in a nutshell*. PrincetonUniversity Press, 2011. (Cited on page 19.)

[78] P. Pons and M. Lapaty. Computing communities in large networks using random walks. (Cited on page 13.)

[79] P. Pons and M. Lapaty. Computing communities in large networks using random walks. 10(2):191–218, 2006. (Cited on page 13.)

[80] Calyampudi Radhakrishna Rao and Sujit Kumar Mitra. Generalized inverse of matrices and its applications. 1971. (Cited on page 7.)

[81] M. Saerens, Y. Achbany, F. Fouss, and L. Yen. Randomized shortest-path problems: Two related models. *Neural Computation*, 21(8):2363–2404, 2009. (Cited on page 18.)

[82] Marco Saerens, Francois Fouss, Luh Yen, and Pierre Dupont. The principal components analysis of a graph, and its relationships to spectral clustering. In *Proceedings of the 15th European Conference on Machine Learning*, ECML'04, pages 371–383, Berlin, Heidelberg, 2004. Springer-Verlag. (Cited on page 12.)

[83] S. Santini and R. Jain. Similarity measures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(9):871–883, Sep 1999. (Cited on page 7.)

[84] B. Scholkopf and A. Smola. Learning with kernels. 2002. (Cited on pages 8 and 9.)

[85] James R Schott. Matrix analysis for statistics. 2005. (Cited on page 7.)

[86] Robert Sedgewick. *Algorithms in C, parts 1-5*. Addison-Wesley, 3rd edition, 1998. (Cited on page 4.)

[87] J. Shawe-Taylor and N. Cristianini. *Kernel methods for pattern analysis*. Cambridge University Press., 2004. (Cited on page 8.)

[88] Henry Small. Co-citation in the scientific literature: A new measure of the relationship between two documents. *Journal of the American Society for Information Science*, 24(4):265–269, 1973. (Cited on page 11.)

[89] A.J. Smola and R. Kondor. Kernels and regularization on graphs. In M. Warmuth and B. Scholkopf, editors, *Learning theory and kernel machines: 16th Annual Conference on Learning Theory and 7th Kernel Workshop, Proceedings (COLT- Kernel 03), volume 2777 of Lecture Notes in Artificial Intelligence*, pages 144–158. Springer, 2003. (Cited on pages 11 and 12.)

[90] A. Subramanya and P. Pratim Talukdar. *Graph-based semi-supervised learning*. Morgan & Claypool Publishers, 2014. (Cited on pages 27 and 58.)

[91] Alireza Tahbaz-Salehi and Ali Jadbabaie. A one-parameter family of distributed consensus algorithms with boundary: From shortest paths to mean hitting times. In *Decision and Control, 2006 45th IEEE Conference on Decision and Control (CDC '06)*, pages 4664–4669. IEEE, 2006. (Cited on page 17.)

[92] H. M. Taylor and S. Karlin. *An introduction to stochastic modeling*. Academic Press, 3rd ed., 1998. (Cited on pages 10 and 18.)

[93] S. Theodoridis and K. Koutroumbas. *Pattern recognition*. Academic Press, 4th ed., 2009. (Cited on pages 7 and 8.)

[94] H. Tong, C. Faloutsos, and Y.-P. Pan. Fast random walk with restart and its applications. In *6th IEEE International Conference on Data Mining (ICDM 06)*, pages 613–622, 2006. (Cited on page 13.)

[95] H. Tong, C. Faloutsos, and Y.-P. Pan. Random walk with restart: Fast solutions and applications. 14(3):327–346, 2008. (Cited on page 13.)

[96] Cédric Villani. Topics in optimal transportation. volume volume 58, Providence, Rhode Island, Etats-Unis, 2003. American Mathematical Society. (Cited on pages 20 and 24.)

[97] Cédric Villani. Optimal transport: old and new. volume olume 338, New York, 2008. Springer, Berlin Heidelberg. (Cited on pages 20 and 24.)

[98] Ulrike von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395 – 416, 2007. (Cited on page 6.)

[99] S. Wasserman and K. Faust. *Social network analysis: Methods and applications*. Cambridge University Press, 1994. (Cited on page 1.)

[100] Frank Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1:80–83, 1945. (Cited on page 34.)

[101] Luh Yen, Marco Saerens, Amin Mantrach, and Masashi Shimbo. A family of dissimilarity measures between nodes generalizing both the shortest-path and the commute-time distances. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, pages 785–793, New York, NY, USA, 2008. ACM. (Cited on pages 14 and 18.)

[102] Dengyong Zhou, Olivier Bousquet, Thomas Navin Lal, Jason Weston, and Bernhard Schölkopf. Learning with local and global consistency. In *Advances in Neural Information Processing Systems 16: Proceedings of the NIPS '03 Conference*, pages 237–244. MIT Press, 2004. (Cited on page 13.)

[103] Xiaojin Zhu and Andrew B Goldberg. *Introduction to semi-supervised learning*. Morgan & Claypool Publishers, 2009. (Cited on pages 27 and 58.)

Annexes

A

A.1. Matlab™ codes

A.1.1. Katz similarity and Leicht's extension

```
function [st] = ...
    Alg_02_02_KatzSimilarityMatrixAndLeichtsExtension(A,alpha)

% Computing the Katz similarity matrix and his Leicht's extension of an
% undirected and weighted graph
%
%   Chapter 2 : Similarity/proximity measures between nodes
%       Section 5 : Global Similarity and Distance Measures
%           Subsection 1 : Katz Index
%
% Arguments :
%
% - A: the n x n adjacency matrix associated to the graph,
% containing affinities
%
% - alpha : the discounting factor (alpha must be between 0 and the
% spectral radius of A)
%
% Returns :
%
% - K_Katz: the Katz similarity matrix
%
% - K_Leicht: the degree-weighted Katz similarity matrix integrating
% contributions from all paths connecting node i and node j, discouting
% paths according to their length
%
% (c) Maxime Duyck

%% Checks of arguments

% Check if squared matrix
[n,m] = size(A);
if(n≠m)
    error('The adjacency matrix is not squared')
end

% Check if symmetric matrix / graph is undirected
for j = 2:n
    for l = 1:j-1
        if(A(j,l) ≠ A(l,j))
            error('The adjacency matrix is not symmetric')
```

```
42          end
43      end
44  end
45
46  % Check if alpha has a correct value
47  spectral_radius = 1/max(eig(A));
48  if(alpha < 0 || alpha > spectral_radius)
49      warning('The spectral radius of A is %d',spectral_radius)
50      error('The discounting factor has not a correct value')
51  end
52
53  %% Algorithm
54  I = eye(n);
55
56  mat = (I-alpha*A)\I;
57  D = diag(A*ones(n,1)); % diagonal degree matrix
58
59  % Katz similarity matrix
60  st.K_Katz = mat - eye(n);
61
62  % Leicht's extension
63  st.K_Leicht = (D\mat)*(D\I);
64  end
```

A.1.2. Commute-time distance and its variants

```
1  function [st] = ...
2      Alg_02_03_ComuteTimeAndEuclideanComuteTimeDistances(A)
3
4  % Computing the (corrected and uncorrected) commute-time and Euclidean
5  % commute-time distances between nodes of an unidrected and weigthed graph
6  %
7  %   Chapter 2 : Similarity/proximity measures between nodes
8  %       Section 5 : Global Similarity and Distance Measures
9  %           Subsection 3 : Commute-Time distance and Euclidean commute-time
10 %           distance
11 %
12 % Arguments :
13 %
14 % - A: the n x n adjacency matrix associated to the graph,
15 % containing affinities
16 %
17 % Returns :
18 %
19 % - Δ_CT: the n x n matrix containing the average commute times between
20 % pairs of nodes
21 %
22 % - Δ_ECT: the n x n matrix containing the square roots of average
23 % commute times between pairs of nodes, i.e., the Euclidean commute-time
24 % distances
25 %
26 % - Δ_CCT: the n x n matrix containing the corrected version of
```

```
% Δ_CT
%
% - Δ_CECT: the n x n matrix containing the corrected version of
% Δ_ECT
%
% (c) Maxime Duyck

%% Checks of arguments

% Check if squared matrix
[n,m] = size(A);
if(n≠m)
    error('The adjacency matrix is not squared')
end

% Check if symmetric matrix / graph is undirected
for j = 2:n
    for l = 1:j-1
        if(A(j,l) ≠ A(l,j))
            error('The adjacency matrix is not symmetric')
        end
    end
end


%% Algorithm

% diagonal degree matrix
e = ones(n,1);
ee = e*e';
I = eye(n);
D = diag(A*e);

% volume of the graph
volume = e'*A*e;

% Laplacian matrix
L = D - A;

% Pseudoinverse of the Laplacian matrix
een = ee/n;
L_plus = ((L+een)\I) + een;

% Average commute-time distance
diago = diag(L_plus);
Δ_CT = volume*(diago*e' + e*diago' - 2*L_plus);
Δ_CT(isnan(Δ_CT))=0;
Δ_CT(isinf(Δ_CT))=100000;

st.Δ_CT = Δ_CT;

% Euclidean commute-time distance
st.Δ_ECT = (Δ_CT).^(.5);

% Corrected commute-time distance
```

```
term = (D \ A) / D; % !!!
dg = diag(term);
Δ_CCT = Δ_CT-volume*(D\ee+ee\D+dg*e'+e*dg'-2*term);
Δ_CCT = Δ_CCT - diag(diag(Δ_CCT)); % set diagonal to zero
st.Δ_CCT = Δ_CCT;

% Corrected Euclidean commute-time distance
st.Δ_CECT = (Δ_CCT).^(.5);

end
```

A.1.3. SimRank similarity

```
function [K] = Alg_02_04_SimRankSimilarityMatrix(A,alpha,tolerance)

% Computing the SimRank similarity matrix between nodes of a directed and
% weighted graph
%
%   Chapter 2 : Similarity/proximity measures between nodes
%       Section 5 : Global Similarity and Distance Measures
%           Subsection 4 : SimRank and an extension for comparing two
%           graphs
%
% Arguments :
%
% - A: the n x n adjacency matrix associated to the graph,
% containing affinities
%
% - alpha: a parameter between 0 and 1 not included
%
% - tolerance : tolerance for convergence of K
%
% Returns :
%
% - K: The n x n similarity matrix
%
% (c) Maxime Duyck

%% Checks of arguments

% default tolerance
if nargin == 2
    tolerance = 1e-3;
end

% Check if squared matrix
[n,m] = size(A);
if(n≠m)
    error('The adjacency matrix is not squared')
end

% Check if alpha has a correct value
```

```
40 if(alpha ≤ 0 || alpha ≥ 1)
41     error('The discounting factor has not a correct value')
42 end
43
44 %% Algorithm
45 % Initialistion K = I
46 K = eye(n);
47
48 % Indegree vector
49 d = A'*ones(n,1);
50
51 % Construction of Q // A little bit different of book algorithm
52 Q = zeros(n);
53 pos_diag = zeros(n);
54 for j = 1:n
55     if(d(j)>0)
56         Q(:,j) = A(:,j)/d(j);
57         pos_diag(j,j) = 1; % Useful for later / 1 if predecessor node,
58                            % 0 otherwise
59     end
60 end
61
62 min_val = 1;
63 while(min_val ≥ tolerance)
64     K_old = K;
65     K = alpha*Q'*A*Q; % Update the similarity matrix
66     K = K - diag(diag(K)) + pos_diag; % set diagonal to 1 if node has
67                                       % a predecessor
68     min_val = min(abs(K-K_old)); % Convergence condition
69 end
70 end
```

A.1.4. Exponential diffusion kernel

```
1  function [st] = Alg_02_06_ExponentialDiffusionKernel(A,alpha)
2
3  % Computing the exponential diffusion kernel and the Laplacian exponential
4  % diffusion kernel matrix of a connected weighted undirected graph
5  %
6  %   Chapter 2 : Similarity/proximity measures between nodes
7  %       Section 6 : Kernel-based similarity measures
8  %           Subsections 1&2 : (Laplacian) Exponential diffusion Kernel
9  %
10 % Arguments :
11 %
12 % - A: the n x n adjacency matrix associated to the graph G_A,
13 % containing affinities
14 %
15 % - alpha : the discounting parameter stricly higher than 0
16 %
17 % Returns :
18 %
```

```
% - K_ED: The n x n exponential diffusion kernel matrix K_ED integrating
% a contribution from all paths connecting node i and node j, discouting
% paths according to their length
%
% - K_LED : the n x n Laplacian exponential diffusion kernel matrix
%
% (c) Maxime Duyck

%% Checks of arguments

% Check if squared matrix
[n,m] = size(A);
if(n≠m)
    error('The adjacency matrix is not squared')
end

% Check if symmetric matrix / graph is undirected
for j = 2:n
    for l = 1:j-1
        if(A(j,l) ≠ A(l,j))
            error('The adjacency matrix is not symmetric')
        end
    end
end

% Check if alpha has a correct value
if(alpha ≤ 0)
    error('The discounting factor has not a correct value')
end

%% Algorithm

% Degree Matrix
D = diag(A*ones(n,1));

% Laplacian Matrix
L = D-A;

% Exponential diffusion kernel matrix
st.K_ED = expm(alpha*A);

% Laplacian Exponential diffusion kernel matrix
st.K_LED = expm(-alpha*L);

end
```

A.1.5. Modified regularized Laplacian kernel

```
function [K_MRL] = Alg_02_07_ModifiedRegularizedLaplacian(A,alpha,gamma)

% Computing the modified regularized Laplacian kernel matrix of a weighted
% undirected graph
```

```
5  %
6  %   Chapter 2 : Similarity/proximity measures between nodes
7  %       Section 6 : Kernel-based similarity measures
8  %           Subsections 3 : Regularized Laplacian kernel and variants
9  %
10 % Arguments :
11 %
12 % - A: the n x n adjacency matrix associated to the graph,
13 % containing affinities
14 %
15 % - alpha : the discounting parameter stricly higher than 0
16 %
17 % - gamma : the parameter controlling importance and relatedness, must be
18 % between 0 and 1 (provides the regularized Laplacian kernel if gamma = 1)
19 %
20 % Returns :
21 %
22 % - K_MRL: The n x n modified regularized Laplacian kernel matrix
23 %
24 % (c) Maxime Duyck
25
26 %% Checks of arguments
27
28 % Check if squared matrix
29 [n,m] = size(A);
30 I = eye(n);
31
32 if(n≠m)
33     error('The adjacency matrix is not squared')
34 end
35
36 % Check if symmetric matrix / graph is undirected
37 for j = 2:n
38     for l = 1:j-1
39         if(A(j,l) ≠ A(l,j))
40             error('The adjacency matrix is not symmetric')
41         end
42     end
43 end
44
45 % Check if alpha has a correct value
46 if(alpha ≤ 0)
47     error('The discounting factor has not a correct value')
48 end
49
50 % Check if gamma has a correct value
51 if(gamma < 0 || gamma > 1)
52     error('the parameter controlling importance has not a correct value')
53 end
54
55 %% Algorithm
56
57 % Degree matrix
58 D = diag(A*ones(n,1));
59
```

```
% Modified Laplacian Matrix
L_gamma = gamma*D - A ;

% the modified regularized Laplacian kernel matrix
K_MRL = (I+alpha*L_gamma)\I;

end
```

A.1.6. Commute-time kernel

```
function [st] = Alg_02_08_CommuteTimeKernelMatrix(A)

% Computing the commute-time kernel matrix of a weighted undirected graph
% (both the usual form and the corrected form)
%
%   Chapter 2 : Similarity/proximity measures between nodes
%       Section 6 : Kernel-based similarity measures
%           Subsections 4 : Commute-time or resistance-distance kernel
%
% Arguments :
%
% - A: the n x n adjacency matrix associated to the graph,
% containing affinities
%
% Returns :
%
% - K_CT: The n x n commute-time kernel matrix
%
% - K_CCT : The n x n corrected version of the comuute-time kernel matrix
%
% (c) Maxime Duyck

%% Checks of arguments

% Check if squared matrix
[n,m] = size(A);
if(n≠m)
    error('The adjacency matrix is not squared')
end

% Check if symmetric matrix / graph is undirected
for j = 2:n
    for l = 1:j-1
        if(A(j,l) ≠ A(l,j))
            error('The adjacency matrix is not symmetric')
        end
    end
end


%% Algorithm

```

```
43 e = ones(n,1);
44 I = eye(n);
45
46 % Degree matrix
47 D = diag(A*e);
48
49 % Degree vector
50 d = D*e;
51
52 %  Laplacian Matrix
53 L = D - A ;
54
55 % Pseudoinverse of the Laplacian matrix
56 een = (e*e')/n;
57 L_plus = ((L+een)\I)-een;
58
59 % Commute-time kernel matrix
60 st.K_CT = L_plus;
61
62 % Compute the centering matrix
63 H = I-een;
64
65 mat = A - ((d*d')/(e'*A*e));
66 sqrt_D = D^(-0.5);
67 M = sqrt_D*mat*sqrt_D;
68
69 % Corrected commute-time kernel matrix
70 st.K_CCT = H*sqrt_D*M*((I-M)\I)*M*sqrt_D;
71
72 end
```

A.1.7. Regularized commute-time kernel

```
1  function [st] = Alg_02_09_RegularizedCommuteTimeKernel(A,alpha)
2
3  % Computing the regularized commute-time kernel and the random walk with
4  % restart similarity of a weighted undirected graph
5  %
6  %   Chapter 2 : Similarity/proximity measures between nodes
7  %       Section 6 : Kernel-based similarity measures
8  %           Subsections 5 : Similarities based on diffusion models:
9  %           regularized commute-time kernel and random walk with
10 %           restart similarity
11 %
12 % Arguments :
13 %
14 % - A: the n x n adjacency matrix associated to the graph,
15 % containing affinities
16 %
17 % - alpha : the discounting parameter between 0 and 1 not included
18 %
19 % Returns :
```

```
20 %
21 % - K_RCT: The n x n regularized commute-time kernel matrix
22 %
23 % - K_RWR : The n x n random walk with restart similarity matrix
24 %
25 % (c) Maxime Duyck
26
27 %% Checks of arguments
28
29 % Check if squared matrix
30 [n,m] = size(A);
31 if(n≠m)
32     error('The adjacency matrix is not squared')
33 end
34
35 % Check if symmetric matrix / graph is undirected
36 for j = 2:n
37     for l = 1:j-1
38         if(A(j,l) ≠ A(l,j))
39             error('The adjacency matrix is not symmetric')
40         end
41     end
42 end
43
44 % Check if alpha has a correct value
45 if(alpha ≤ 0 || alpha≥1)
46     error('The discounting factor has not a correct value')
47 end
48
49 %% Algorithm
50
51 % Degree matrix
52 D = diag(A*ones(n,1));
53
54 % Regularized commute-time kernel matrix
55 st.K_RCT = (D-alpha*A)\I;
56
57 % Random walk with restart similarity matrix
58 st.K_RWR = (D-alpha*A)\D;
59
60 end
```

A.1.8. Markov diffusion square distance and its variants

```
1 function [st] = ...
2     Alg_02_10_MarkovDiffusionSquareDistance(A,w,t)
3
4 % Computing the Markov diffusion square distance and kernel matrix of a
5 % weighted directed graph
6 %
7 %   Chapter 2 : Similarity/proximity measures between nodes
8 %       Section 6 : Kernel-based similarity measures
```

```
%           Subsections 6 : Markov diffusion distance and kernel
%
% Arguments :
%
% - A: the n x n adjacency matrix associated to the graph,
% containing affinities
%
% - w : the weighting factors w_k ≥ 0 associated with each node k in
% vector w. When the Markov chain is regular, the usual weighting factor
% is w_k = 1/pi_k where pi_k is entry k of the stationary distribution
%
% - t : the considered number of steps (or transitions) of the Markov
% process
%
% Returns :
%
% - Δ_2MD: The n x n original version of Markov diffusion squared
% distances matrix
%
% - Δ_2MDA: The n x n time-averaged version Markov diffusion squared
% distances matrix
%
% - K_2MD: The n x n original version of Markov diffusion kernel matrix
%
% - K_2MDA : The n x n time-averaged version of Markov diffusion kernel
% matrix
%
% (c) Maxime Duyck

%% Checks of arguments

% Check if squared matrix
[n,m] = size(A);
if(n≠m)
    error('The adjacency matrix is not squared')
end

% Check if length of w is correct
if(length(w)≠n)
    error('The weigthing has not the correct dimension')
end

% Check if alpha has a correct value
if (min(w) < 0)
    error('The weighting vector has at least a negative value')
end

%% Algorithm

e = ones(n,1);

% The row-normalization matrix
D = diag(A*e);

% The transition matrix associated with A
```

```
64 P = D\A;
65
66 H = eye(n)-((e*e')/n) ;
67
68 Z = zeros(size(P));
69 for i=1:t
70     Z = Z + P^i;
71 end
72 Z = Z/t;
73
74 % The diagonal matrix containing the weights
75 D_w = diag(w);
76
77 % Original version of Markov diffusion squared distances matrix
78 st.Δ_2MD = diag((P^t)*D_w*(P^t)')*e' + e*(diag((P^t)*D_w*(P^t)'))'...
79      - 2*(P^t)*D_w*(P^t)';
80
81
82 % Time-averaged version of Markov diffusion squared distances matrix
83 st.Δ_2MDA = diag(Z*D_w*Z')*e' + e*(diag(Z*D_w*Z'))'...
84      - 2*Z*D_w*Z';
85
86 % Original version of centered Markov difusion kernel matrix
87 st.K_2MD = H*(P^t)*D_w*(P^t)'*H;
88
89 % Time-averaged version of centered Markov difusion kernel matrix
90 st.K_2MDA = H*Z*D_w*Z'*H;
91
92 end
```

A.1.9. Logarithmic forest distance

```
1  function [K] = Alg_03_01_LogarithmicForestDistanceMatrix(A,alpha)
2
3  % Computing the logarithmic forest distance matrix between nodes of an
4  % undirected graph
5  %
6  %   Chapter 3 : *Families of dissimilarity between nodes
7  %       Section 3 : The p-resistance distance
8  %
9  % Arguments :
10 %
11 % - A: the n x n adjacency matrix associated to the graph,
12 % containing affinities
13 %
14 % - alpha: a positive parameter
15 %
16 % Returns :
17 %
18 % - K: The n x n logarithmic forest distance matrix
19 %
20 % (c) Maxime Duyck
```

```
21
22 %% Checks of arguments
23
24 % Check if squared matrix
25 [n,m] = size(A);
26 if(n≠m)
27     error('The adjacency matrix is not squared')
28 end
29
30 % Check if alpha has a correct value
31 if(alpha ≤ 0)
32     error('The discounting factor has not a correct value')
33 end
34
35 % Check if symmetric matrix / graph is undirected
36 for j = 2:n
37     for l = 1:j-1
38         if(A(j,l) ≠ A(l,j))
39             error('The adjacency matrix is not symmetric')
40         end
41     end
42 end
43
44 %% Algorithm
45 % Initialisation
46 e = ones(n,1);
47 I = eye(n);
48
49 % The degree matrix
50 D = diag(A*e);
51
52 % The Laplacian matrix
53 L = D-A;
54
55 % Regularized Laplacian Kernel Matrix
56 K_rl = (I + alpha*L)\I;
57
58 % Logarithmic transformation of K_rl
59 if alpha == 1
60     S = log(K_rl);
61 else
62     S = (alpha-1)*(log(K_rl)/log(alpha));
63 end
64
65 % Logarithmic forest distance matrix
66 K = diag(S)*e' + e*(diag(S))' - 2*S;
67 end
```

A.1.10. Regular bag-of-paths

```
1 function [Pi] = Alg_03_02_RegularBagOfPathsProbabilityMatrix(A,theta,C)
2
```

```
3  % Computing the regular bag-of-paths probability matrix of a weighted and
4  % and directed graph
5  %
6  %   Chapter 3 : *Families of dissimilarity between nodes
7  %       Section 4 : *Bag-of-paths framework
8  %           Subsection 4 : Computing the probability of sampling a path
9  %                          starting in i and ending in j
10 %
11 % Arguments :
12 %
13 % - A: the n x n adjacency matrix associated to the graph,
14 % containing affinities
15 %
16 % - C: the n x n cost matrix associated to the graph,
17 %
18 % - tetha: the inverse temperature parameter
19 %
20 % Returns :
21 %
22 % - Pi: The n x n bag of paths probability matrix containing the
23 % probability of drawing a path starting in node i and ending in node j
24 % from a bag of paths, when sampling paths according to a Gibbs-Boltzmann
25 % distribution.
26 %
27 % (c) Maxime Duyck
28
29 %% Checks of arguments
30
31 % Check if squared matrix
32 [n,m] = size(A);
33 if(n≠m)
34     error('The adjacency matrix is not squared')
35 end
36
37 if nargin == 2
38     C = ComputationOfCostMatrix(A);
39 end
40
41 %% Algorithm
42 % Initialisation
43 e = ones(n,1);
44 I = eye(n);
45
46 % The degree matrix
47 D = diag(A*e);
48
49 % The reference transition probability matrix
50 P_ref = D\A;
51
52 % Elementwise and exponential multiplication
53 W = P_ref .* exp(-theta*C);
54
55 % The fundamental matrix
56 Z = (I-W)\I;
57
```

```
% Normalization factor - the partition function
z_dotdot = e'*Z*e;

% Regular bag-of-paths probability matrix
Pi = Z/z_dotdot;
end
```

A.1.11. Bag-of-hitting-paths

```
function [st] = ...
    Alg_03_03_BagOfHittingPathsProbabilityMatrix(A,theta,C)
% Computing the bag of hitting paths probability matrix of a weighted and
% and directed graph
%
%   Chapter 3 : *Families of dissimilarity between nodes
%       Section 4 : *Bag-of-paths framework
%           Subsection 5 : Bag of hitting paths model
%
% Arguments :
%
% - A: the n x n adjacency matrix associated to the graph,
% containing affinities
%
% - C: the n x n cost matrix associated to the graph,
%
% - tetha: the inverse temperature parameter
%
% Returns :
%
% - Pi_h:The n x n bag of hitting paths probability matrix with zero length
% paths included containing the probability of drawing a path starting in
% node i and ending in node j from a bag of paths, when sampling paths
% according to a Gibbs-Boltzmann distribution.
%
% - Pi_bh: The n x n bag of hitting paths probability matrix with zero
% lenght paths excluded
%
% (c) Maxime Duyck

%% Checks of arguments

% Check if squared matrix
[n,m] = size(A);
if(n≠m)
    error('The adjacency matrix is not squared')
end

if nargin == 2
    C = ComputationOfCostMatrix(A);
end

%% Algorithm
```

```
44 % Initialisation
45 e = ones(n,1);
46 I = eye(n);
47
48 % The degree matrix
49 D = diag(A*e);
50
51 % The reference transition probability matrix
52 P_ref = D\A;
53
54 % Elementwise and exponential multiplication
55 W = P_ref .* exp(-theta*C);
56
57 % The fundamental matrix
58 Z = (I-W)\I;
59
60 % The column-normalization matrix for the bag of hitting paths
61 % probabilities
62 D_h = diag(diag(Z));
63
64 % Column-normalize the fundamental matrix
65 Z_h = Z*(D_h\I);
66
67 % Compute normalization factor - the partition function
68 Z_hh = e'*Z_h*e;
69
70 % The bag of hitting paths probability matrix with zero paths included
71 st.Pi_h = Z_h/Z_hh;
72
73 % Column-normalize the fundamental matrix and substract zero-length
74 % path contributions
75 Z_bh = Z_h - I;
76
77 % Compute normalization factor - the partition function
78 Z_bhh = e'*Z_bh*e;
79
80 % The bag of hitting paths probability matrix with zero paths excluded
81 st.Pi_bh = Z_bh/Z_bhh;
82
83 end
```

A.1.12. Bag-of-paths surprisal distance

```
1 function [Δ_h] = ...
2     Alg_03_04_BagOfHittingPathsSurprisalDistanceMatrix(Pi_h)
3 % Computing the bag of hitting paths probability matrix of a weighted and
4 % and directed graph
5 %
6 %   Chapter 3 : *Families of dissimilarity between nodes
7 %       Section 5 : *Three distance measures based on the
8 %                   bag of hitting paths probabilities
9 %           Subsection 1 : First distance based on the associated
```

```
%                          surprisal measure
%
% Arguments :
%
% - Pi_h:The n x n hitting paths probability matrix, associated to a
% weighted and directed graph, containing the bag of hitting paths
% probabilities (included zero-length paths). To compute it, use the
% algorithm 3.3
%
% Returns :
%
% - Δ_h :The n x n bag of hitting paths surprisal distance matrix
% containing the pairwise distances between nodes.
%
% (c) Maxime Duyck

%% Checks of arguments

% Check if squared matrix
[n,m] = size(Pi_h);
if(n≠m)
    error('The adjacency matrix is not squared')
end

%% Algorithm
% Take elementwise logarithm
Δ_h = log(Pi_h);

% Symmetrize the matrix
Δ_h = .5*(Δ_h + Δ_h');

% Put diagonal to zero
Δ_h = Δ_h - diag(diag(Δ_h));

end
```

A.1.13. Free-energy distance

```
function [Δ_phi] = ...
    Alg_03_05_BagOfHittingPathsPontentialOrFreeEnergyDistanceMatrix(A,theta,C)
% Computing the bag of hitting paths potential, or free energy, distance
% matrix of a weighted and directed graph
%
%   Chapter 3 : *Families of dissimilarity between nodes
%       Section 5 : *Three distance measures based on the
%                    bag of hitting paths probabilities
%           Subsection 2 : Second distance based on the bag hitting paths
%
% Arguments :
%
% - A: the n x n adjacency matrix associated to the graph,
% containing affinities
```

```
15 %
16 % - C: the n x n cost matrix associated to the graph,
17 %
18 % - tetha: the inverse temperature parameter
19 %
20 % Returns :
21 %
22 % - Δ_phi : the n x n bag of hitting paths potential distance matrix
23 %   containing the pairwise distances between nodes.
24 %
25 % (c) Maxime Duyck
26
27 %% Checks of arguments
28
29 % Check if squared matrix
30 [n,m] = size(A);
31 if(n≠m)
32     error('The adjacency matrix is not squared')
33 end
34
35 if nargin == 2
36     C = ComputationOfCostMatrix(A);
37 end
38
39 %% Algorithm
40 % Initialisation
41 e = ones(n,1);
42 I = eye(n);
43
44 % The row normalization matrix
45 D = diag(A*e);
46
47 % The reference transition probability matrix
48 P_ref = (D\I)*A;
49 % Elemental exponential and multiplication
50 W = P_ref .* exp(-theta*C);
51
52 % The fundamental matrix
53 Z = (I-W)\I;
54
55 % The column-normalization matrix for hitting probabilities
56 D_h = diag(Z);
57
58 % Column-normalize the fundamental matrix
59 Z_h = Z*(D\I);
60
61 % Take elementwise logarithm for computing the potentials
62 Phi = -log(Z_h)/theta;
63
64 % Symmetrize the matrix
65 Δ_phi = .5*(Phi + Phi');
66
67 % Put diagonal to zero
68 Δ_phi = Δ_phi - diag(diag(Δ_phi));
69 end
```

A.1.14. Randomized shortest-path dissimilarity

```
function [Δ_rsp] = ...
    Alg_03_06_RandomizedShortestPathDissimilarityMatrix(A,theta,C)
% Computing randomized shortest path dissimilarity matrix for hitting paths
% of a weighted and directed graph
%
%   Chapter 3 : *Families of dissimilarity between nodes
%       Section 6 : *Randomized shortest path dissimilarity and
%                    the free energy distance
%           Subsection 1 : Randomized shortest path dissimilarity
%
% Arguments :
%
% - A: the n x n adjacency matrix associated to the graph,
% containing affinities
%
% - C: the n x n cost matrix associated to the graph,
%
% - tetha: the inverse temperature parameter
%
% Returns :
%
% - Δ_phi : the n x n randomized shortest path dissimilarity matrix
%   containing the pairwise distances between nodes.
%
% (c) Maxime Duyck

%% Checks of arguments

% Check if squared matrix
[n,m] = size(A);
if(n≠m)
    error('The adjacency matrix is not squared')
end

if nargin == 2
    C = ComputationOfCostMatrix(A);
end

%% Algorithm
% Initialisation
e = ones(n,1);
I = eye(n);

% The row normalization matrix
D = diag(A*e);

% The reference transition probability matrix
P_ref = (D\I)*A;

% Elemental exponential and multiplication
W = P_ref .* exp(-theta*C);
```

```
52
53 % The fundamental matrix
54 Z = (I-W)\I;
55
56 % The expected costs for nonhitting paths
57 S = (Z*(C.*W)*Z)/Z;
58
59 % The randomized shortest path dissimilarity matrix for hitting paths
60 Δ_rsp = .5*(S + S' - e*(diag(S))' - diag(S)*e');
61
62 end
```

A.1.15. Bag-of-paths absorption probabilities

```
1  function [B] = ...
2      Alg_03_07_BagOfPathsAbsorptionProbabilitiesOfAbsorbingNodes(A,theta,abs,C)
3  % Computing the bag-of-paths absorption probabilities to a set of absorbing
4  % nodes of a weighted and directed graph
5  %
6  %   Chapter 3 : *Families of dissimilarity between nodes
7  %       Section 7 : *Bag-of-Paths absoption probabilities
8  %           Subsection 1 :Computing the bag-of-Paths absoption
9  %                         probabilities
10 %
11 % Arguments :
12 %
13 % - A: the n x n adjacency matrix associated to the graph,
14 % containing affinities
15 %
16 % - C: the n x n cost matrix associated to the graph,
17 %
18 % - tetha: the inverse temperature parameter
19 %
20 % - abs : the set of absorbing nodes (n x 1 vector)
21 %         abs(i) == 1 if i is an absorbing node
22 %         abs(i) == 0 else
23 %
24 % Returns :
25 %
26 % - B : the (n - |abs| x |abs|) matrix containing the bag-of-paths
27 % absoption probabilities for each transient node
28 %
29 % (c) Maxime Duyck
30
31 %% Checks of arguments
32
33 % Check if squared matrix
34 [n,m] = size(A);
35 if(n≠m)
36     error('The adjacency matrix is not squared')
37 end
38
```

```
39 if nargin == 3
40     C = ComputationOfCostMatrix(A);
41 end
42
43 if nargin == 2
44     C = ComputationOfCostMatrix(A);
45     abs = ComputationOfAbsorbingNodes(A);
46 end
47
48 k = length(abs);
49 if(n≠k)
50     error('The set of absorbing nodes do not contain all nodes')
51 end
52
53 % Check if binary vector
54 for i=1:n
55     if (abs(i) ≠ 1 && abs(i) ≠ 0)
56         error('The set of absorbing nodes is not binary')
57     end
58 end
59
60
61
62 %% Algorithm
63 % Initialisation
64 e = ones(n,1);
65 I = eye(n);
66
67 % The set of transient nodes (T = V\abs)
68 %T = ones(n,1) - abs;
69 %for i=1:n
70 %     if (abs(i) == 1)
71 %         T(i) = 0;
72 %     else
73 %         T(i) = 1;
74 %     end
75 %end
76
77
78 % The row normalization matrix
79 D = diag(A*e);
80
81 % The reference transition probability matrix
82 P_ref = (D\I)*A;
83
84 % Elemental exponential and multiplication
85 W_a = P_ref .* exp(-theta*C);
86
87 % Set rows corresponding to absorbing nodes to zero
88 for i=1:n
89     if abs(i) == 1
90         W_a(i,:) = 0;
91     end
92 end
93
```

```
94  % The fundamental matrix
95  Z_a = (I-W_a)\I;
96
97  % Extract sub-matrix whose rows correspond to transient nodes and columns
98  % to absorbing nodes
99  k = n - sum(abs);
100 l = sum(abs);
101
102 if(l≠0)
103   B_1 = zeros(k,n);
104   B = zeros(k,l);
105
106   index = 1; % Delete of rows not corresponding to transient nodes
107   for i = 1:n
108       if (abs(i) == 0)
109           B_1(index,:) = Z_a(i,:);
110          index = index+1;
111       end
112   end
113
114   index = 1; % Delete of column not corresponding to absorbing nodes
115   for i = 1:n
116       if (abs(i) == 1)
117           B(:,index) = B_1(:,i);
118           index = index+1;
119       end
120   end
121 elseif(l==0)
122   B = Z_a;
123 end
124
125 % Compute normalization factor for each row
126 e_1 = ones(k,1);
127 I_1 = eye(k);
128 D_B = diag(B*e_1);
129
130 % Normalize the matrix for computing absoption probabilities
131 B = (D_B\I_1)*B;
132 end
```

A.1.16. Bag-of-paths covariance measure

```
1  function [K_BoP] = ...
2      Alg_03_08_BagOfPathsCovarianceMatrixBetweenNodes(theta,C,P_ref)
3  % Computing the bag-of-paths absorption probabilities to a set of absorbing
4  % nodes of a weighted and directed graph
5  %
6  %   Chapter 3 : *Families of dissimilarity between nodes
7  %       Section 7 : *Bag-of-Paths absoption probabilities
8  %           Subsection 1 :Computing the bag-of-Paths absoption
9  %                         probabilities
10 %
```

```
% Arguments :
%
% If 2 inputs :
% - theta: the parameter controlling the degree of randomness
%
% - C: the n x n adjacency matrix associated to the graph,
% containing affinities
%
% If 3 inputs :
% - theta: the parameter controlling the degree of randomness
%
% - C: the n x n cost matrix associated to the graph,
%
% - P_ref: the n x n reference transition probabilities matrix
%
% Returns :
%
% - K_BoP : the bag of paths covariance matrix between pairs of nodes
% containing the elements k_BoP(k,l) = cov(k,l)
%
% (c) Maxime Duyck

%% Checks of arguments

% Initialisation
[n,m] = size(C);
e = ones(n,1);
I = eye(n);

if nargin == 2
    A = C;
    C = ComputationOfCostMatrix(A);
    D = diag(A*e);
    P_ref = (D\I)*A;
end

% Check if squared matrix
if(n~=m)
    error('The adjacency matrix is not squared')
end



%% Algorithm
% Elemental exponential and multiplication
W = P_ref .* exp(-theta*C);

% The fundamental matrix
Z = (I-W)\I;

z_dotk = sum(Z);
z_kdot = sum(Z,2);
z_dotdot = sum(sum(Z));

% The partition function
```

```
Z_bar = z_dotdot - n;

K_BoP = zeros(n,n);
for k=1:n
    for l=k:n
        if (l==k)
            Δ = 1;
        else
            Δ = 0;
        end
        p_1 = (z_dotk(k)-1)*z_kdot(k)*Δ;
        p_2 = z_dotk(k)*(z_kdot(l)-1)*(Z(l,k)-Δ);
        p_3 = z_kdot(l)*(z_dotk(k)-1)*(Z(k,l)-Δ);
        p_4 = (1/Z_bar)*(z_kdot(k)*z_kdot(l)*(z_dotk(k)-1)*(z_dotk(l)-1));

        K_BoP(k,l) = (1/Z_bar)*(p_1 + p_2 + p_3 - p_4);


        K_BoP(l,k) = K_BoP(k,l);
    end
end

end
```

A.1.17. Randomized optimal transport

```
function [D] = rand_OT(A,betaP,hitting)

if nargin == 2
  hitting = true;
end

% Initialization
[n,m] = size(A);
sIn = repmat(1/n,n,1);
sOut = repmat(1/n,n,1);

persist=0.1;
convThres= 1e-10;

myMin = 1e-30;
myMax = intmax('int64')/1000;

I = eye(n);
e = ones(n,1);

% Cost matrix
C = ComputationOfCostMatrix(A);

% Probability matrix
D = diag(A*e);
P = (D\I)*A;
```

```
27
28 % Supply and demand
29  sIn = sIn / sum(sIn);
30  sOut = sOut / sum(sOut);
31
32 if hitting == true
33   W = P .* exp(-betaP*C);
34   Z = (I-W)\I;
35   D_h = diag(diag(Z));
36   Z = Z*(D_h\I);
37   outVal = sOut;
38   numerator = e;
39 else
40   Q = I - P';
41   pinvQ = pinv(Q);
42   q = (I - pinvQ * Q);
43   q = q(:,1);
44   nRef0 = pinvQ * (sIn - P' * sOut);
45   gammaP = max((sOut - nRef0) ./ q) + persist;
46   nRef = nRef0 + gammaP * q;
47   alpha = sOut ./ nRef;
48   Pmod = (I - diag(alpha)) * P;
49   % matrices W and Z, vectors divisor and outVal
50   W = exp(-betaP * C) .* Pmod;
51   Z = (I-W)\I;
52   outVal = alpha;
53   numerator = nRef;
54 end
55
56 % Iterations
57  mIn = e;
58  mOut = e;
59  convergence = false;
60  FE = myMax;
61
62 while convergence != true
63   FEPrev = FE;
64   mIn = e ./ (Z * (mOut .* outVal));
65   mOut = numerator ./ (Z' * (mIn .* sIn));
66
67   % log gives sometimes complex numbers
68   FE = real(sum(-log(mIn) .* sIn) + sum(-log(mOut) .* sOut));
69   maxDiff = abs(FEPrev - FE);
70   if maxDiff < convThres
71     convergence = true;
72   end
73 end
74
75 % Coupling
76 Pi = diag(mIn .* sIn) * Z * diag(mOut .* outVal);
77
78 % Dissimilarity
79 D = real(-(log(Pi) + log(Pi'))/2);
80 D = D- diag(diag(D));
81
```

```
82 % Security
83 D(D > myMax/(n*(n-1))) = myMax/(n*(n-1));
84 D(isinf(D)) = myMax/(n*(n-1));
85
86 D = D./ mean(D(D>0));
87
88 end
```

Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve **www.uclouvain.be/epl**