A novel way of computing dissimilarities between nodes of a graph, with application to collaborative filtering and subspace projection of the graph nodes

Francois Fouss*, Alain Pirotte†, Jean-Michel Renders† & Marco Saerens†

January 31, 2006

Abstract

This work presents a new perspective on characterizing the similarity between elements of a database or, more generally, nodes of a weighted, undirected, graph. It is based on a Markov-chain model of random walk through the database. More precisely, we compute quantities (the **average commute time**, the **pseudoinverse of the Laplacian matrix** of the graph, etc) that provide similarities between any pair of nodes, having the nice property of increasing when the number of paths connecting those elements increases and when the "length" of paths decreases. It turns out that the square root of the average commute time is a Euclidean distance and that the pseudoinverse of the Laplacian matrix is a kernel (it contains inner-products closely related to commute times). A procedure for computing the subspace projection of the node vectors of the graph that preserves as much variance as possible in terms of the commute-time distance – a principal components analysis (PCA) of the graph – is also introduced. This graph PCA provides a nice interpretation to the "**Fiedler vector**", widely used for graph partitioning. The model is evaluated on a collaborative-recommendation task where suggestions are made about which movies people should watch based upon what they watched in the past. Experimental results on the MovieLens database show that the Laplacian-based similarities perform well in comparison with other methods. The model, which nicely fits into the so-called "statistical relational learning" framework, could also be used to compute document or word similarities, and, more generally, could be applied to machine-learning and pattern-recognition tasks involving a database.

*François Fouss, Alain Pirotte and Marco Saerens are with the Information Systems Research Unit (ISYS), IAG, Université catholique de Louvain, Place des Doyens 1, B-1348 Louvain-la-Neuve, Belgium. Email: {saerens, pirotte, fouss}@isys.ucl.ac.be.

†Jean-Michel Renders is with the Xerox Research Center Europe, Chemin de Maupertuis 6, 38240 Meylan (Grenoble), France. Email: jean-michel.renders@xrce.xerox.com.

1. Introduction

1.1. General introduction

Exploiting the graph structure of large repositories, such as digital documents repositories, the web environment, or large databases in general, is relevant to many areas of computer science. For instance, Kleinberg's suggestion to distinguish web pages that are hubs and authorities (see [41]; called the HITS algorithm) has been well-received in the community (for a review, see [2]).

This work views a database as a collection of element sets (tables) connected by relationships. The model exploits the graph structure of the database to compute a similarity measure between elements (the work could have been presented as computing dissimilarities instead, and the word "proximities" used as a substitute for either). All the developments in this paper are valid in the more general context of computing similarities between nodes of a **weighted**, **undirected**, **graph**.

Computing similarities between pairs of elements allows, for instance, to determine the item that is most relevant (similar) to a given item. Also, elements in a set can be assigned a category provided by elements from another set. Computing similarities between elements of the same set amounts to a clustering task.

For example, imagine a simple movie database with three sets of elements `people`, `movie`, and `movie_category`, and two relationships `has_watched`, between `people` and `movie`, and `belongs_to`, between `movie` and `movie_category`.

- Computing similarities between people allows to cluster them into groups with similar interest about watched movies.
- Computing similarities between people and movies allows to suggest movies to watch or not to watch.
- Computing similarities between people and movie categories allows to attach a most relevant category to each person.

The procedure used to compute similarities is based on a Markov-chain model. We define a **random walk** through the database by assigning a transition probability to each link. Thus, a random walker can jump from element to element and each element therefore represents a state of the Markov chain.

The **average first-passage time** $m(k|i)$ (see for instance [39]) is the average number of steps needed by a random walker for reaching state k for the first time, when starting from state i. The symmetrized quantity $n(i,j) = m(j|i)+m(i|j)$, called the **average commute time** (see for instance [26]), provides a distance measure between any pair of states. The fact that this quantity is indeed a distance on a graph was proved independently by Klein & Randic [40] and Gobel & Jagers [26].

We will see later that $[n(i,j)]^{1/2}$, which is also a distance between nodes, takes a remarkable form; it will be called the **Euclidean Commute Time Distance** (ECTD). We also introduce the **average first-passage cost** $o(k|i)$ which generalizes the average first-passage time by assigning a cost to each transition.

Another quantity of interest, closely related to the ECTD, that will play a fundamental role, is the **pseudoinverse of the Laplacian matrix** of the graph ($\mathbf{L}^{+}$).

Indeed, it turns out that $\mathbf{L}^{+}$ contains the inner products of the node vectors in a Euclidean space preserving the ECTD (a Euclidean space where the nodes are exactly separated by ECTD). It can therefore be considered as a similarity measure between nodes and it is a valid kernel (a Gram matrix, see for instance [57]).

All these quantities have the nice property of decreasing (or increasing, depending on whether the quantity is a dissimilarity or a similarity measure) when the number of paths connecting two elements increases and when the "length" of any path decreases (the communication is facilitated). In short, two elements are considered similar if there are many short paths connecting them. The "shortest path" or "geodesic" distance does not have the nice property of decreasing when connections between nodes are added: it does not capture the fact that strongly connected nodes are at a smaller distance than weakly connected nodes. With a few notable exceptions (see the related work below), while being interesting alternatives to the well-known "shortest path" distance on a graph [10], those quantities have not yet been fully exploited in the context of collaborative recommendation or, more generally, in pattern recognition and machine learning. This fact has, however, already been recognized in the field of mathematical chemistry where there were attempts to use the "commute time" distance instead of the "shortest path" distance [40].

1.2. Some related work

In [14] and [16], Chebotarev & Shamis proposed a similarity measure between nodes of a graph integrating indirect paths, based on the matrix-forest theorem. Interestingly enough, this quantity is also related to the Laplacian matrix of the graph. While the authors prove some nice properties about this similarity measure, no experiment investigating the effectiveness of this quantity is performed. We will therefore investigate this matrix-forest based measure, together with the other introduced quantities, in the experimental section (a precise description of this model can be found in Section 7).

Some authors recently considered similarity measures based on random-walk models. For instance, Harel & Koren [29] investigated the possibility of clustering data according to some random-walk related quantities, such as the probability of visiting a node before returning to the starting node. They showed through clustering experiments that their algorithm is able to cluster arbitrary non-convex shapes. White & Smyth [64], independently of our work, investigated the use of the average first-passage time as a similarity measure between nodes. Their purpose was to generalize the random-walk approach of Page, Brin & Winograd [48] by capturing a concept of "relative centrality" of a given node with respect to some other node of interest.

More recently, Newman [45] suggested to use a random-walk model in order to compute the "betweenness centrality" of a given node in a graph. Newman counts how often a node is traversed during a random walk between two other nodes (a couple of nodes). Then, this quantity is averaged over every couple of nodes, providing a general measure of betweenness [63] associated to each node.

Still another approach has been investigated by Faloutsos et al. [23] who extract the "most relevant" connected subgraph relating two nodes of interest, using a method based on electrical flow. Moreover, in [49], Palmer & Faloutsos define a similarity function between categorical attributes, called "refined escape probability", based on random walks and electrical networks concepts. They show that this quantity provides a reasonably good measure for clustering and classifying categorical attributes. In a recent paper [44], Nadler et al. proposed a dissimilarity measure between nodes of a graph based on a continuous-time diffusion process. They show that there are some interesting links between their model and our approach. We derived the discrete-time counterpart of their model, and are currently investigating its performance.

Very recently, Brand [8] proposes the use of various quantities derived from the commute time for collaborative recommendation. He shows, as we do, that angular-based quantities perform much better than the commute time because the commute time is quite sensible to the node degree. These conclusions confirm our experimental results, as will be shown in the experimental section.

Notice finally that our approach based on a random-walk model on a graph is closely related to spectral clustering and spectral embedding techniques (for a recent account, see [20]), as detailed in [55]. Random-walk models on a graph also proved useful in the context of learning from labeled and unlabeled data; see for instance [65].

1.3. Main contributions

In summary, in addition to suggesting quantities for computing similarities between nodes of a graph, this paper has four main contributions, three more theoretical and one more experimental:

1. We show that all the introduced quantities can be expressed in closed form in terms of the pseudoinverse of the Laplacian matrix of the graph. This generalizes results obtained by Klein & Randic [40], derived for the ECTD only, based on the electrical equivalence. Since the pseudoinverse of the Laplacian matrix plays a key role and has a nice interpretation in terms of random walk on a graph, we prove some of its properties.

2. We show that the pseudoinverse of the Laplacian matrix of the graph is a valid kernel (a Gram matrix; see for instance [57]). It therefore defines a kernel on a graph and can be interpreted as a similarity measure.

3. We show that we can project the nodes space of the graph into a Euclidean subspace that approximately preserves the ECTD. This subspace is optimal in the following sense: it keeps as much variance of the projected data as possible (in terms of the ECTD). It is therefore an equivalent of principal components analysis (PCA) and classical multidimensional scaling (MDS), in terms of the ECTD. This provides a nice interpretation to the "Fiedler vector", widely used in graph partitioning. Finally, we also show that the ECTD PCA can be viewed as a special regularized Laplacian kernel, as introduced by Smola & Kondor [61].

4. From an experimental point of view, we show that these quantities can be used in the context of collaborative recommendation. For a recent account of collaborative recommendation techniques, see [18], [30] or [33]. Indeed, all the introduced concepts are illustrated on a collaborative recommendation task where movies are suggested for people to watch from a database of previously watched movies. In particular the inner-product based quantities involving the Laplacian matrix provide good, stable, results. We suggest that the same model

could be used to compute document or word similarities, and, more generally, be applied to other pattern-recognition and machine-learning tasks involving a database.

Our objective here is not to develop a state-of-the-art collaborative recommendation system; rather, the application to collaborative recommendation aims to illustrate the interest of Markov-based similarity measures to nontrivial database-mining tasks. This approach fits quite naturally into the so-called "multi-relational data mining" and the "statistical relational learning" frameworks (see for instance [22] and the other papers in the same issue of the SIGKDD newsletter).

1.4. Outline of the paper

The present work is an extended and improved follow-up of two previously published papers: [55], introducing the theoretical model based a random-walk model, and [25], containing preliminary experimental results obtained on the collaborative recommendation task. In the present work, all the proofs are included, computational issues are discussed, and the experiments have been highly expanded.

The paper is structured as follows. Section 2 introduces the random-walk model. Section 3 develops our similarity measures as well as the iterative formulae to compute them. Section 5 shows how the average first-passage time, the average first-passage cost, and the average commute time can be computed in closed form from the pseudoinverse of the Laplacian matrix of the graph. It also derives a number of interesting properties of the Laplacian pseudoinverse. Section 6 introduces a subspace projection of the nodes of the graph that maximizes the variance of the projected data. Section 7 specifies our experimental methodology. Section 8 illustrates the concepts with experimental results obtained on the MovieLens database. Section 9 is the conclusion.

2. A Markov-chain model of database navigation

2.1. Definition of the weighted graph

A weighted graph G is associated with a database in the following obvious way: database elements correspond to nodes of the graph and database links correspond to edges.

In our movie example, each element of the `people`, `movie`, and `movie_category` sets corresponds to a node of the graph, and each `has_watched` and `belongs_to` link is expressed as an edge.

The weight w_{ij} of the edge connecting node i and node j should be set to some meaningful value, with the following convention: the more important the relation between node i and node j, the larger the value of w_{ij}, and consequently the easier the communication through the edge. We require the weights to be both positive ($w_{ij} > 0$) and symmetric ($w_{ij} = w_{ji}$). For instance, for an `has_watched` edge, the weight could be set to the number of times that the person watched the corresponding movie. The elements a_{ij} of the symmetric adjacency matrix $\mathbf{A}$ of the graph are defined as usual as: $a_{ij} = w_{ij}$ if node i is connected to node j, and $a_{ij} = 0$ otherwise.

Because of the way the graph is defined, people who watch the same kind of movies, and therefore have similar taste, will be connected by a comparatively large number of short paths. On the contrary, for people with different interests, there will be fewer paths connecting them and these paths will be longer.

2.2. A random-walk model on the graph

The Markov chain describing the sequence of nodes visited by a random walker is called a random walk. A random variable $s(t)$ contains the current state of the Markov chain at time step t: if the random walker is in state i at time t, then $s(t) = i$.

The random walk is defined with the following single-step transition probabilities of jumping from any state or node $i = s(t)$ to an adjacent node $j = s(t+1)$: $\mathrm{P}(s(t+1) = j | s(t) = i) = a_{ij}/a_{i.} = p_{ij}$, where $a_{i.} = \sum_{j=1}^{n} a_{ij}$.

The transition probabilities depend only on the current state and not on the past ones (first-order Markov chain). Since the graph is connected, the Markov chain is irreducible, that is, every state can be reached from any other state. If this is not the case, the Markov chain can be decomposed into closed subsets of states which are independent (there is no communication between them), each closed subset being irreducible, and the procedure can be applied independently on these closed subsets.

If we denote the probability of being in state i at time t by $\pi_i(t) = \mathrm{P}(s(t) = i)$ and we define $\mathbf{P}$ as the transition matrix with entries $p_{ij} = \mathrm{P}(s(t+1) = j | s(t) = i)$, the evolution of the Markov chain is characterized by $\boldsymbol{\pi}(t+1) = \mathbf{P}^{\mathrm{T}} \boldsymbol{\pi}(t)$, with $\boldsymbol{\pi}(0) = \boldsymbol{\pi}^0$ and where T is the matrix transpose. This provides the state probability distribution $\boldsymbol{\pi}(t) = [\pi_1(t), \pi_2(t), ..., \pi_n(t)]^{\mathrm{T}}$ at time t once the initial probability density $\boldsymbol{\pi}^0$ is known. It is well-known (see [53]) that such a Markov chain of random walk on a graph is time-reversible ($\pi_i p_{ij} = \pi_j p_{ji}$) with stationary probabilities given by

$$\pi_i = \frac{\sum\limits_{j=1}^{n} a_{ij}}{\sum\limits_{i,j=1}^{n} a_{ij}} = \frac{a_{i.}}{a_{..}}$$

This value is the probability of finding the Markov chain in state $s = i$ in the long-run behavior.

This has important implications. For instance, it is known that all the eigenvalues of the transition matrix of a reversible Markov chain are real and distinct (see for instance [9]).

For more details on Markov chains, the reader is invited to consult standard textbooks on the subject (e.g., [39], [47]).

3. Probability of absorption, average first-passage time/cost and average commute time

This section reviews three basic quantities that can be computed from the definition of the Markov chain, that is, from its transition probability matrix: the probability of absorption, the average first-passage time, and the average commute time. We also introduce the average first-passage cost which generalizes the average first-passage time. Relationships allowing to compute these quantities are simply introduced without proof (see, e.g., [39] or [47] for a more formal treatment).

3.1. The probability of absorption

The **probability of absorption** $u(k|i)$ is the probability that a random walker starting from some initial state i enters for the first time state $k \in S^a$ (S^a is a subset of states) before reaching any other state belonging to S^a. The states of the set S^a are "absorbing states" in the sense that, once the random walker reaches one of them, it stops walking. Thus, once an absorbing state has been reached, the probability of staying in it is 1. A recurrence relation allowing to compute the probability of absorption can be obtained by elementary probability theory (see for instance [62]; a proof is provided in Appendix A):

$$\begin{cases} u(k|i) = p_{ik} + \displaystyle\sum_{j=1,\, j\neq k}^{n} p_{ij}\, u(k|j), \text{ for } i \notin S^a \text{ and } k \in S^a \\ u(k|k) = 1, \text{ for } k \in S^a \\ u(k|i) = 0, \text{ for } i, k \in S^a \text{ and } i \neq k \end{cases} \tag{3.1}$$

This system of linear equations can be solved by iterating the relations (the relaxation method, see [21]). The probability of absorption can also be obtained by algorithms that were developed in the Markov-chain community (see for instance [39]), or by using the pseudoinverse of the Laplacian matrix of the graph (see Section 5).

3.2. The average first-passage time and average first-passage cost

The **average first-passage time** $m(k|i)$ is defined as the average number of steps that a random walker, starting in state $i \neq k$, will take to enter state k for the first time [47]. More precisely, we define the minimum time until hitting state k, when starting from state i, as $T_{ik} = \min\,(t \geq 0 \mid s(t) = k \text{ and } s(0) = i)$ for one realization of the stochastic process. The random walker will pass through k repeatedly; The minimum time corresponds to its first passage. The average first-passage time is the expectation of this quantity: $m(k|i) = E[T_{ik}|s(0) = i]$.

In a similar way, the **average first-passage cost** $o(k|i)$ is the average cost incurred by the random walker starting from state i to reach state k for the first time. The cost of each transition is given by $c(j|i)$ (a cost matrix) for any states i, j. Notice that $m(k|i)$ is obtained as a special case where $c(j|i) = 1$ for all i, j.

The recurrence relations for computing $m(k|i)$ and $o(k|i)$ can easily be obtained by first-step analysis (see [39], [47]):

$$\begin{cases} m(k|k) = 0 \\ m(k|i) = 1 + \displaystyle\sum_{j=1,\, j\neq k}^{n} p_{ij}\, m(k|j), \text{ for } i \neq k \end{cases} \tag{3.2}$$

$$\begin{cases} o(k|k) = 0 \\ o(k|i) = \displaystyle\sum_{j=1}^{n} p_{ij}\, c(j|i) + \sum_{j=1,\, j\neq k}^{n} p_{ij}\, o(k|j), \text{ for } i \neq k \end{cases} \tag{3.3}$$

The meaning of these formulae is quite obvious: to go from state i to state k, first go to any adjacent state j and proceed from there. These quantities can be computed by iterating these recurrence relations, by using some dedicated algorithms developed in the Markov-chain community (see for instance [39]) or by using the pseudoinverse of the Laplacian matrix of the graph, as shown in this paper (see Section 5).

3.3. The average commute time

A closely related quantity, the **average commute time** $n(i,j)$ is defined as the average number of steps that a random walker, starting in state $i \neq j$, will take to enter state j for the first time and go back to i. That is, $n(i,j) = m(j|i) + m(i|j)$. Notice that, while $n(i,j)$ is symmetric by definition, $m(i|j)$ is not.

As shown by several authors [26], [40], the average commute time is a distance measure since, for any states i, j, k: (1) $n(i,j) \geq 0$, (2) $n(i,j) = 0$ if and only if $i = j$, (3) $n(i,j) = n(j,i)$, and (4) $n(i,j) \leq n(i,k) + n(k,j)$. It will be referred to as the "**commute-time distance**". Because of a close relationship between the random-walk model and electrical networks theory, this distance is also called "**resistance distance**". Indeed, $n(i,j)$ is proportional to the effective resistance between node i and node j of the corresponding network, where a resistance w_{ij}^{-1} is assigned to each edge. We will show in Section 5 that $[n(i,j)]^{1/2}$, which is also a distance on the graph, takes a remarkable form.

As already mentioned, the commute-time distance between two points has the desirable property of decreasing when the number of paths connecting the two points increases and when the length of paths decreases (see Section 4 for a proof based on electrical networks theory). The usual **shortest-path distance** (also called geodesic distance) does not have this property: the shortest-path distance does not capture the fact that strongly connected nodes are closer than weakly connected nodes.

Methods for computing the quantities defined above are based on algorithms introduced in the Markov-chain community or on iterative procedures (Equation 3.2). For instance, Kemeny & Snell proposed a general method in the appendix of their book [39] (see also [34] or [50]).

4. Relations to electrical networks

There is an intriguing correspondence between random walk on a graph and electrical networks theory, as popularized by Doyle and Snell in their nice book [21]. Probability of absorption and average commute time both have equivalents in terms of electrical networks.

4.1. Definition of the electrical network

We view our weighted graph as an electrical network where the weights on edges represent **conductances**. Conductances are defined as the inverse of resistances: $c_{ij} = 1/r_{ij}$. In other words, we define an electrical network with conductances $c_{ij} = w_{ij}$.

The main quantities of interest are the **potential** v_i, defined at each node of the network, as well as the **current**, i_{ij}, defined on each edge. If we denote by $N(i)$ the set of adjacent nodes of node i, the fundamental equations relating these basic quantities are

$$i_{ij} = c_{ij}(v_i - v_j) \tag{4.1}$$

$$\sum_{i \in N(k)} i_{ki} = I_k \tag{4.2}$$

where I_k is the source of current at node k.

Another quantity of fundamental importance is the **effective resistance**, r^e_{ij}. Suppose we impose a potential difference V between nodes i and j: a potential $v_i = V$ is established at node i while $v_j = 0$ at node j. A current $I_i = \sum_{k \in N(i)} i_{ik}$ will flow into the network from source i (we assume that current flows from higher potential to lower potential) to sink j. The amount of current that flows depends upon the overall resistance in the network. The effective resistance between i and j is defined by $r^e_{ij} = (v_i - v_j)/I_i = V/I_i$. The reciprocal quantity $c^e_{ij} = 1/r^e_{ij}$ is the **effective conductance**.

In Appendix B, we show how the potential at any node of the network can be computed from the Laplacian matrix of the graph, in the case of a single source and a single sink of current.

$$v_i - v_j = I(\mathbf{e}_i - \mathbf{e}_j)^{\mathrm{T}} \mathbf{L}^+ (\mathbf{e}_a - \mathbf{e}_b)$$

4.2. Electrical equivalent to average commute time and probability of absorption

In [13], it is shown that $n(i,j) = V_G\, r^e_{ij}$, where V_G is the volume of the graph, $V_G = \sum_{i,j=1}^n a_{ij}$. In other words, average commute time and effective resistance basically measure the same quantity; this quantity is therefore also called **resistance distance** – in the sequel, we will indifferently use resistance distance or commute time distance.

The probability of absorption $u(k|i)$ also has an electrical equivalent. Remember that $u(k|i)$ is the probability that a random walker starting from some initial state i enters for the first time state $k \in S^a$ (S^a is the set of absorbing states) before reaching any other state belonging to S^a. Results obtained by Doyle and Snell [21] show that $u(k|i)$ represents the potential at node i (v_i) when a unit voltage is applied at node k ($v_k = 1$), while the potential at any other "absorbing" node $\in S^a$ is set to 0 ($v_j = 0$ for $j \in S^a$, $j \neq k$).

5. Computation of the basic quantities by means of $\mathbf{L}^+$

In this section, we show how formulae for computing the average first-passage time, the average first-passage cost, and the average commute time can be derived from Equations 3.2 and 3.3, by using the Moore-Penrose pseudoinverse of the Laplacian matrix of the graph ($\mathbf{L}^+$), which plays a fundamental role and has a number of interesting properties. The developments in this section are inspired by the work of Klein & Randic [40] who proved, based on the electrical equivalence, that the effective resistance (the equivalent of the average commute time) can be computed from the Laplacian matrix. We extend their results by showing how the formula computing the average commute time in terms of $\mathbf{L}^+$ can be directly derived from Equation 3.2, and by providing formulae for the average first-passage time and the average first-passage cost.

5.1. The pseudoinverse of the Laplacian matrix

The Laplacian matrix $\mathbf{L}$ of the graph is defined in the usual manner, $\mathbf{L} = \mathbf{D} - \mathbf{A}$, where $\mathbf{D} = \text{diag}(a_{i.})$ with $d_{ii} = [\mathbf{D}]_{ii} = a_{i.} = \sum_{j=1}^{n} a_{ij}$ (element i, j of $\mathbf{D}$ is $[\mathbf{D}]_{ij} = d_{ij}$), if there are n nodes in total.

We suppose that the graph is connected, that is, any node can be reached from any other node. In this case, $\mathbf{L}$ has rank $n-1$ [19]. If $\mathbf{e}$ is a column vector made of 1's (i.e., $\mathbf{e} = [1, 1, ..., 1]^{\text{T}}$, where T denotes the matrix transpose) and $\mathbf{0}$ is a column vector made of 0's, $\mathbf{Le} = \mathbf{0}$ and $\mathbf{e}^{\text{T}}\mathbf{L} = \mathbf{0}^{\text{T}}$ hold: $\mathbf{L}$ is doubly centered. The null space of $\mathbf{L}$ is therefore the one-dimensional space spanned by $\mathbf{e}$. Moreover, one can easily show that $\mathbf{L}$ is symmetric and positive semidefinite (see for instance [19]).

Let us denote by l_{ij} the $[\mathbf{L}]_{ij}$ element of the Laplacian matrix $\mathbf{L}$. The Moore-Penrose pseudoinverse of $\mathbf{L}$ (see for instance [3]) will be denoted by $\mathbf{L}^{+}$, with elements $l_{ij}^{+} = [\mathbf{L}^{+}]_{ij}$. The concept of pseudoinverse generalizes the matrix inverse to matrices which are not of full rank, or even rectangular. It provides closed-form solutions to systems of linear equations for which there is no exact solution (in which case it provides a solution in the least-square sense) or when there is an infinity of solutions. A thorough treatment of matrix pseudoinverses and their applications can be found in [6].

In Appendix C, we prove some useful properties of $\mathbf{L}^{+}$; in particular, we show that: (1) $\mathbf{L}^{+}$ is symmetric, (2) If $(\lambda_i \neq 0, \mathbf{u}_i)$ are (eigenvalues, eigenvectors) of $\mathbf{L}$, then $(\lambda_i^{-1} \neq 0, \mathbf{u}_i)$ are corresponding (eigenvalues, eigenvectors) of $\mathbf{L}^{+}$; on the other hand, if $(\lambda_j = 0, \mathbf{u}_j)$ are (eigenvalues, eigenvectors) of $\mathbf{L}$, then they are also (eigenvalues, eigenvectors) of $\mathbf{L}^{+}$, (3) $\mathbf{L}^{+}$ is doubly centered, and (4) $\mathbf{L}^{+}$ is positive semidefinite. Moreover it can be shown that $\mathbf{L}^{+}$ can be computed with the following formula (see [52], chapter 10) involving a standard matrix inversion:

$$\mathbf{L}^{+} = \left(\mathbf{L} - \frac{\mathbf{e}\mathbf{e}^{\text{T}}}{n} \right)^{-1} + \frac{\mathbf{e}\mathbf{e}^{\text{T}}}{n} \tag{5.1}$$

where n is the number of nodes.

5.2. The probability of absorption

Appendix D details the computation of the probability of absorption by state a before absorption by state b, when starting from state k, for a Markov model with two absorbing states a and b:

$$u(a|k) = \frac{l_{ka}^{+} - l_{kb}^{+} - l_{ab}^{+} + l_{bb}^{+}}{l_{aa}^{+} + l_{bb}^{+} - 2l_{ab}^{+}} \tag{5.2}$$

$$u(b|k) = \frac{l_{kb}^{+} - l_{ka}^{+} - l_{ab}^{+} + l_{aa}^{+}}{l_{aa}^{+} + l_{bb}^{+} - 2l_{ab}^{+}} \tag{5.3}$$

5.3. The average first-passage time/cost

Appendix E shows that the computation of the average first-passage time in terms of $\mathbf{L}^+$ can be obtained from Equation 3.2:

$$m(k|i) = \sum_{j=1}^{n} \left(l^+_{ij} - l^+_{ik} - l^+_{kj} + l^+_{kk} \right) d_{jj} \tag{5.4}$$

where d_{ii} is the sum of the weights at node i, $d_{ii} = \sum_{j=1}^{n} a_{ij} = a_{i.}$.

A similar formula (see Appendix E) is derived from Equation 3.3 for the average first-passage cost in terms of $\mathbf{L}^+$:

$$o(k|i) = \sum_{j=1}^{n} \left(l^+_{ij} - l^+_{ik} - l^+_{kj} + l^+_{kk} \right) b_j \tag{5.5}$$

where $b_i = \sum_{j=1}^{n} a_{ij} c(j|i)$.

5.4. The average commute time

Appendix F shows that the computation of the average commute time in terms of $\mathbf{L}^+$ is easily derived from Equation 5.4:

$$n(i,j) = V_G \left(l^+_{ii} + l^+_{jj} - 2l^+_{ij} \right) \tag{5.6}$$

where V_G is the volume of the graph ($V_G = \sum_{k=1}^{n} d_{kk}$). This formula has already been obtained by using the electrical equivalent of the average commute time (the effective resistance) in [40]; see also Section4.

If we define $\mathbf{e}_i$ as the ith column of $\mathbf{I}$, $\mathbf{e}_i = [\underset{1}{0}, \ldots, \underset{i-1}{0}, \underset{i}{1}, \underset{i+1}{0}, \ldots, \underset{n}{0}]^{\mathrm{T}}$, we obtain the remarkable form:

$$n(i,j) = V_G \, (\mathbf{e}_i - \mathbf{e}_j)^{\mathrm{T}} \mathbf{L}^+ (\mathbf{e}_i - \mathbf{e}_j) \tag{5.7}$$

where each node i is represented by a unit vector, $\mathbf{e}_i$, in the node space (the space spanned by $\{\mathbf{e}_i\}$).

We easily observe that $[n(i,j)]^{1/2}$ is a distance in the Euclidean space of the nodes of the graph since $\mathbf{L}^+$ is positive semidefinite. It will therefore be called the **Euclidean Commute Time Distance** (ECTD). This is nothing else than a Mahalanobis distance with a weighting matrix $\mathbf{L}^+$.

If the matrices are too large, the computation based on the pseudoinverse becomes impractical; in this case, one may use iterative techniques based on Equations 3.1, 3.3, 3.2 and on the sparseness of the transition probability matrix.

5.5. Computational issues

If the number of nodes is too large, the direct computation based on the pseudoinverse becomes intractable; in this case, one may use iterative techniques based on Equations 3.2 or 3.3 and on the sparseness of the transition probabilities matrix [28], [54].

Another alternative that proved useful for computing $\mathbf{L}^+$ for sparse graphs is first to compute a sparse Cholesky factorization of the Laplacian matrix, $\mathbf{L} = \mathbf{R}\mathbf{R}^{\mathrm{T}}$ (see [17] for computing the Cholesky factorization of positive semidefinite matrices). If $\mathbf{L}$ is sparse, the lower-triangular factor $\mathbf{R}$ is sparse as well, although less sparse than the original matrix $\mathbf{L}$. It is therefore useful to first compute a permutation of the original matrix in order to obtain a "band" matrix. Once the factorization has been computed, the ith column of $\mathbf{L}^+$ can be calculated by the following three steps (see [31], pp. 440-441):

1. Compute the projection of the basis vector $\mathbf{e}_i$ on the column space of $\mathbf{L}$; this projection is provided by $\mathbf{y}_i = \mathrm{proj}_{\mathbf{L}}(\mathbf{e}_i) = (\mathbf{I} - \mathbf{e}\mathbf{e}^{\mathrm{T}}/n)\mathbf{e}_i$.
2. Find one solution $\mathbf{x}'_i$ of $\mathbf{L}\mathbf{x} = \mathbf{R}\mathbf{R}^{\mathrm{T}}\mathbf{x} = \mathbf{y}_i$ by back-substitution ($\mathbf{R}$ is lower-triangular and sparse).
3. Project $\mathbf{x}'_i$ on the row space of $\mathbf{L}$ (which is the same as the column space since $\mathbf{L}$ is symmetric), $\mathbf{x}^*_i = \mathrm{proj}_{\mathbf{L}}(\mathbf{e}_i) = (\mathbf{I} - \mathbf{e}\mathbf{e}^{\mathrm{T}}/n)\mathbf{x}'_i$. $\mathbf{x}^*_i$ corresponds to the ith column of $\mathbf{L}^+$ [31].

This procedure allows to compute the columns of $\mathbf{L}^+$ "on demand".

Still another viable approach, based on a truncated series expansion, is proposed by Brand in [8]. The special case of a bipartite graph (as the movie database) is developed in [32], where the authors propose an optimized method for computing the pseudoinverse of the Laplacian matrix in this situation.

6. Maximum variance subspace projection of the node vectors

6.1. Mapping to a Euclidean space preserving the ECTD

Based on the eigenvector decomposition of $\mathbf{L}^+$, the node vectors $\mathbf{e}_i$ can be mapped into a new Euclidean space that preserves the ECTD (see Appendix H): $n(i,j) = V_G\,(\mathbf{x}'_i - \mathbf{x}'_j)^{\mathrm{T}}(\mathbf{x}'_i - \mathbf{x}'_j) = V_G\,||\mathbf{x}'_i - \mathbf{x}'_j||^2$ where we made the transformations $\mathbf{e}_i = \mathbf{U}\mathbf{x}_i$ or $\mathbf{x}_i = \mathbf{U}^{\mathrm{T}}\mathbf{e}_i$, $\mathbf{x}'_i = \mathbf{\Lambda}^{1/2}\mathbf{x}_i$, and where $\mathbf{U}$ is an orthonormal matrix made of the eigenvectors of $\mathbf{L}^+$ (ordered in decreasing order of corresponding eigenvalues λ_k) and $\mathbf{\Lambda} = \mathrm{diag}(\lambda_k)$. So, in this new n-dimensional Euclidean space, the transformed node vectors $\mathbf{x}'_i$ are **exactly separated** (according to the standard Euclidean distance) by ECTD.

In Appendix I, it is shown that the $\mathbf{x}'_i$ are centered and that the pseudoinverse of the Laplacian matrix contains the **inner products** of the transformed node vectors, $l^+_{ij} = \mathbf{x}'^{\mathrm{T}}_i\mathbf{x}'_j$. Therefore, $\mathbf{L}^+$ is a **kernel** (a Gram matrix) and can be considered as a **similarity matrix** between the nodes (as in the vector-space model in information retrieval). It therefore defines a new kernel on a graph, like the von Neumann kernel [58], the diffusion kernel [42], and the recently introduced regularized Laplacian kernel [35], [61]. In fact, it can easily be shown that the $\mathbf{L}^+$ kernel can be obtained from the regularized Laplacian kernel by using a special regularization operator (see the end of the next subsection). This result is worth mentioning since once we have defined a meaningful kernel on a graph, a number of interesting measures are provided almost for free (kernel PCA, etc.; see for instance [57]). We are currently comparing various other well-defined kernels on a graph on the same collaborative recommendation task; this will be the subject of a subsequent paper.

One key issue here is to assess whether the **distance-based measures** (for instance the ECTD) or the **inner-product based measures** (for instance $\mathbf{L}^+$) perform best in the context of collaborative recommendation. It is well-known that in the context of the vector-space model of information retrieval, inner-product based measures outperform Euclidean distances when computing proximities between documents [1]. In the present case, ECTD are Euclidean distances, while $\mathbf{L}^+$ contains the inner products between the node vectors. In this framework, another measure of interest is the **cosine** between node vectors, which is defined as

$$\cos^+(i,j) = l^+_{ij}/\sqrt{l^+_{ii} l^+_{jj}} \tag{6.1}$$

6.2. Subspace projection of the node vectors (the principal components analysis of a graph)

$\mathbf{L}^+$ can be approximated by retaining only the $m < (n-1)$ first eigenvectors (the smallest eigenvalue is 0) of its eigenvector decomposition:

$$\widetilde{\mathbf{L}}^+ = \sum_{k=1}^{m<n-1} \lambda_k \, \mathbf{u}_k \mathbf{u}_k^{\mathrm{T}} \tag{6.2}$$

where the $\mathbf{u}_k$ are the eigenvectors of $\mathbf{L}^+$ and λ_k the corresponding eigenvalues (see Appendix H for details). A new transformation of the node vectors is therefore defined by

$$\begin{cases} \widetilde{\mathbf{x}}_i = \widetilde{\mathbf{U}}^{\mathrm{T}} \mathbf{e}_i \\ \widetilde{\mathbf{x}}'_i = \widetilde{\mathbf{\Lambda}}^{1/2} \widetilde{\mathbf{x}}_i \end{cases} \tag{6.3}$$

where $\widetilde{\mathbf{U}} = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \quad \mathbf{u}_m, \mathbf{0}, \ldots, \mathbf{0}]$ and $\widetilde{\mathbf{\Lambda}} = \mathrm{diag}[\lambda_1, \lambda_2, \ldots, \lambda_m, 0, \ldots, 0]$. The $\widetilde{\mathbf{x}}'_i$ are column vectors containing zeroes from position $m{+}1$ on: $\widetilde{\mathbf{x}}' = [\widetilde{x}'_1, \widetilde{x}'_2, ..., \widetilde{x}'_m, 0, ..., 0]^{\mathrm{T}}$. This subspace is thus a m-dimensional space where the ECTD are approximately preserved. A bound on this approximation is provided in Appendix J: $||n(i,j) - \widetilde{n}(i,j)|| \leq V_G \sum_{k=m+1}^{n-1} \lambda_k$.

This decomposition is similar to Principal Components Analysis (PCA) in the sense that the projection of the node vectors in this subspace has maximal variance (in terms of ECTD) among all the possible candidate projections (see [55]; see also Appendix K). It can be shown that it has a number of interesting links with both spectral clustering (see for instance Shi and Malik [59] or Ding [20]; details are provided in [55]) and spectral embedding [4], [5]; in particular, the ECTD PCA provides a natural interpretation for both spectral clustering and embedding.

Moreover, it is easy to show that performing a multidimensional scaling on the ECTD gives exactly the same results as the principal components analysis. Indeed, classical multidimensional scaling [7] amounts to performing the spectral decomposition of the matrix given by $-(1/2)$ $\mathbf{JNJ}$ and to retaining the first m eigenvalues and eigenvectors, where $\mathbf{J}$ is the centering matrix ($\mathbf{J} = \mathbf{I} - \mathbf{e}\mathbf{e}^{\mathrm{T}}/n$) and $\mathbf{N}$ is the ECTD distance matrix ($[\mathbf{N}]_{ij} = n(i,j)$). It is not difficult to show that $-(1/2)$ $\mathbf{JNJ}$ is nothing else than $\mathbf{L}^+$ (times V_G), so that both formulations are equivalent [7].

As for PCA, we expect that the first few principal components contain most of the information about the basic structure of the graph; the next components related to small eigenvalues being related to noise. If this assumption is true, we expect to obtain better performance by keeping only a few components of the PCA. Of course, keeping only two or three components allows to visualize the graph.

Since both $\mathbf{L}$ and $\mathbf{L}^+$ have rank $(n-1)$ and have the same set of eigenvectors but inverse eigenvalues, we do not need to explicitly compute the pseudoinverse of $\mathbf{L}$ in order to find the projection. We only need to compute the m smallest (except $\lambda_n = 0 = \lambda_n^+$) eigenvectors (that is, with lowest eigenvalue) of $\mathbf{L}$, which become the largest of $\mathbf{L}^+$.

This last result shows that the $\widetilde{\mathbf{L}}^+$ similarity matrix is a regularized Laplacian kernel (as defined in [42]) with a regularization factor given by $r(\lambda_i) = \lambda_i^{-1}$ for the smallest m (non-null) eigenvalues λ_i of the Laplacian matrix and $r(\lambda_i) = 0$ for the remaining eigenvalues. It trivially penalizes the largest eigenvalues of the Laplacian matrix, by "cutting" them off in the regularized Laplacian kernel.

Finally, notice that this graph PCA provides a nice interpretation to the "**Fiedler vector**" [24], [43], widely used for graph partitioning [12], [51]: the Fiedler vector simply contains the projections of the node vectors on the first principal component of the graph, $\mathbf{u}_1$. Indeed, the Fiedler vector corresponds to the smallest nonzero eigenvalue of the Laplacian matrix, which is also the first eigenvector $\mathbf{u}_1$ of $\mathbf{L}^+$.

7. Experimental methodology

Remember that each element of the `people` and the `movie` sets corresponds to a node of the graph. Each node of the `people` set is connected by an edge to each movie watched by the corresponding person. Our experiments do not take the `movie_category` set into account in order to perform fair comparisons between the different scoring algorithms. Indeed, three standard scoring algorithms (i.e., maximum frequency, cosine and nearest-neighbours algorithms) cannot naturally use the `movie_category` set to rank the movies.

7.1. Data set

Our experiments were performed on a real movie database from the web-based recommender system MovieLens (www.movielens.umn.edu). We used a sample of this database proposed in [56]: enough persons (i.e., 943 persons) were randomly selected to obtain 100,000 ratings (considering only persons that had rated 20 or more movies on a total of 1682 movies).

The experiments were divided into two parts. A preliminary experiment was performed in order to tune the parameters of several scoring algorithms (we do not show these results in this paper), while the main experiment aims to further assess the scoring algorithms by cross-validation (see Section 8).

The **preliminary experiment** allowed to optimize the parameters of various scoring algorithms (see Section 7.4), namely parameter α for the Katz method, the number of dimensions for PCA CT, the similarity measure for the k nearest neighbours algorithm, and the number of neighbours k for each of the scoring algorithm when using the indirect method (see Section 7.3). The database was divided into a `training set` and a `test set`, by splitting the data as suggested in [56]. The `test set` contains exactly 10 ratings for each of the 943 persons (about $10,000$ ratings), while the `training set` contains the remainder of the ratings (about $90,000$

ratings).

In the **main experiment**, the data set was divided into n subsets ($n = 10$), and the scoring algorithm was applied n times (10-fold cross-validation). Each time, one of the n subsets was used as the `test set` while the other $n - 1$ subsets were merged into a `training set`. Then the average result across all n trials was computed. Notice that no link between a specific person and a specific movie belongs both to the `training set` and to the `test set` and that, for each person, the `test set` may contain any number of movies.

The results shown here do not take into account the numerical value of the ratings provided by the persons but only the fact that a person has or has not watched a movie (i.e., entries in the person-movie matrix are 0's and 1's).

For each person, each scoring algorithm described in Section 7.4 extracts from the `training set` a list of preferences about movies, computed as similarities (scores) between people nodes and movie nodes. From that information, we take out a ranked list of all the movies *that the person has not watched*, according to the `training set`. In that list, the movies closest to the person, in terms of the similarity score, are considered the most relevant.

7.2. Performance evaluation

To evaluate the scoring algorithms described in Section 7.4, we compared their performance using three measures of performance: (1) the degree of agreement (which is a variant of Somers'D [60]), (2) a percentile score, and (3) a recall score. For each person, the `test set` contains a set of movies that the person has actually watched and that have been removed from the `training set`, i.e., these links have been deleted from the graph. Those movies are part of the ranked list supplied by each scoring algorithm which contains all the movies that the person has not watched, according to the `training set`.

7.2.1. Degree of agreement

To compute the degree of agreement, we consider each pair of movies such that where one movie is in the `test set` for that person and the other movie is not in the `test set`. A scoring algorithm ranks the pair in the correct order if, in the ranked list, the movie from the `test set` (a movie that was actually watched) precedes the movie not in the `test set` (a movie that was actually not watched). The degree of agreement is thus the proportion (in %) of pairs ranked in the correct order with respect to the total number of pairs [60]. The idea here is that the scoring algorithm should favour the movies that have indeed been watched (those belonging to the `test set`) by ranking them first.

The results discussed in the next section compute a single degree of agreement for all the persons. A degree of agreement of 50% (50% of all the pairs are in correct order and 50% are in bad order) is equivalent to a random ranking. A degree of agreement of 100% means that the proposed ranking is identical to the ideal ranking (the watched movies are always ranked first).

7.2.2. Percentile

The percentile score is simply the average (on all persons) of the position (in %) of the median movie from the `test set`, with respect to the whole set of ranked movies. For instance, if the `test set` contains three movies, ranked in 10th, 15th and 40th position, and there are 100 ranked movies in total, the percentile score will be 15%. A random ranking would provide a percentile score around 50%. This measure should be as low as possible for good performance (near 0% for a perfect model).

7.2.3. Recall

The recall score is the average (on all persons) of the proportion (in %) of movies from the `test set` that appear among the *top* n of the ranked list, for some given n. A recall score of 100% indicates that the scoring algorithm always positioned the movies in the `test set` among the *top* n of the ranked list. This measure should be as high as possible for good performance, with perfect recall obtained at 100%. We will report the recall for the *top* 10 and the *top* 20 movies (for a total of 1682 movies).

7.3. Direct versus indirect method

There are three ways to determine which movies to suggest to a particular person based on computing similarities between pairs of nodes:

- **Direct method**: use each scoring algorithm to compute the similarities between a given person and all the movies. The movie on the top of the list is suggested first to the given person.
- **User-based indirect method**: first, use each scoring algorithm as in the direct method to compute the similarities between a given person p_0 and all the other persons p; then, for the given person, compute from its k nearest neighbours (in the present case, nearest neighbours are persons) the predicted value of each movie. The predicted value of person p_0 for movie m_0 is computed as a sum, weighted by $\text{sim}(p_0, p)$, of the values (the link weight, 0 or 1) of movie m_0 for the k closest persons p of person p_0:

$$\text{pred}(p_0, m_0) = \frac{\sum_{p=1}^{k} \text{sim}(p_0, p)\ a_{pm_0}}{\sum_{p=1}^{k} \text{sim}(p_0, p)} \tag{7.1}$$

  where a_{pm_0} is 1 if person p watched the movie m_0 and 0 otherwise. The movies that are proposed first to person p_0 are those that have the highest predicted values. For each scoring algorithm, we varied systematically the number of neighbours k $(= 1, 2, ..., 10, 20, ..., 100)$ and we keep the value of k giving the best result. Notice that the value of k depends on the measure used to evaluate the performance (see Section 7.2).
- **Movie-based indirect method**: this corresponds to the methodology proposed by Karypis in its SUGGEST approach [37]. First, use each scoring algorithm as in the direct method to compute the similarities between a given

movie and all the other movies; then, for a given person p_0, compute from the k nearest neighbours of the movies he watched (in the present case, nearest neighbours are movies) the predicted value of each movie. The predicted value of person p_0 for movie m_0 is computed as a sum, weighted by $\text{sim}(m_0, m)$, of the values (the link weight, 0 or 1) of movie m_0 for the k closest movies m of movie m_0:

$$\text{pred}(p_0, m_0) = \frac{\sum_{m=1}^{k} \text{sim}(m_0, m)\ a_{p_0 m}}{\sum_{m=1}^{k} \text{sim}(m_0, m)} \tag{7.2}$$

where $a_{p_0 m}$ is defined as before. The movies that are proposed first to person p_0 are those that have the highest predicted values. As for the user-based indirect method, we optimized the number of neighbours. Remember that this way to suggest movies is equivalent to the SUGGEST method proposed in [37].

Notice that, if the algorithm provides a dissimilarity $\text{dis}(i, j)$, we use $(\max - \text{dis}(i, j))/(\max - \min)$ to convert it into a similarity.

7.4. Scoring algorithms

Twelve scoring, or ranking, algorithms are compared. The person-independent maximum-frequency algorithm (MaxF) will serve as a reference to appreciate the quality of the other scoring algorithms. Six scoring algorithms are based on our random-walk model: the average commute time (normal and PCA-based), the average first-passage time (one-way and return, see later), and the pseudoinverse of the Laplacian matrix (normal and $\cos^+$). The other algorithms are standard techniques: k-nearest neighbours techniques, cosine coefficient, Katz' method, and the shortest-path algorithm. Finally, we also include the random-forest based similarity measure proposed in [14]. We now describe these algorithms in more details.

Maximum-frequency algorithm (MaxF). This scoring algorithm ranks the movies by the number of persons who watched them. Movies are suggested to each person in order of decreasing popularity. The ranking is thus the same for all the persons. MaxF is equivalent to basing the decision only on the a priori probabilities in supervised classification. Notice that the MaxF algorithm can only be used in the direct method.

Average commute time (CT). We use the average commute time $n(i, j)$ to rank the elements of the considered set, where i and j are elements of the database. For instance, if we want to suggest movies to people using the direct method, we compute the average commute time between `people` elements and `movie` elements. The lower the value is, the more similar the two elements are. In the sequel, this quantity will simply be referred to as "*commute time*".

Principal components analysis based on ECTD (PCA CT). In Section 6, we showed that, based on the eigenvector decomposition of $\mathbf{L}^+$, the nodes can be mapped into a new Euclidean space (with more than 2600 dimensions in this case) that preserves the ECTD, or a m-dimensional subspace keeping as much variance as possible, in terms of ECTD.

Thus, after performing a PCA and keeping a given number of principal components, we recompute the distances in this reduced subspace. These approximate

ECTD between people and movies are then used to rank the movies for each person. We varied the dimension m of the subspace from 10 to 2620 by step of 10. The best results were obtained for 60 principal components ($m = 60$).

Average first-passage time (One-way). In a similar way, we use the average first-passage time $m(i|j)$, to compute a similarity score between element i and element j of the database. This quantity will simply be referred to as "*one-way time*".

Average first-passage time (Return). As a similarity between element i and element j of the database, this scoring algorithm uses $m(j|i)$ (the transpose of $m(i|j)$), that is, the average time used to reach j when starting from i. This quantity will simply be referred to as "*return time*".

Pseudoinverse of the Laplacian matrix ($\mathbf{L}^{+}$). $\mathbf{L}^{+}$ provides a similarity measure ($\text{sim}(i, j) = l_{ij}^{+}$) since it is the matrix containing the inner products of the node vectors in the Euclidean space where the nodes are exactly separated by the ECTD. Once we have computed the similarity matrix, movies are ranked according to their similarity with the person. In addition, we used the same procedure as for the "PCA CT": rely on the principal components analysis subspace. Since we did not observe any improvement in comparison with $\mathbf{L}^{+}$ we do not show the results here.

Cosine based on $\mathbf{L}^{+}$($\mathbf{cos}^{+}$). This scoring algorithm computes similarities from Equation 6.1 to rank the movies.

k-nearest neighbours (kNN). The nearest-neighbour method is one of the simplest and oldest methods for performing general classification tasks. It can be represented by the following rule: to classify an unknown pattern, choose the class of the nearest example in the training set as measured by a similarity metric. When choosing the k nearest examples to classify the unknown pattern, one speaks about k-nearest neighbours techniques.

Using a nearest-neighbour technique requires a measure of "closeness" or "similarity". There is often a great deal of subjectivity involved in the choice of a similarity measure (see [36]). Important considerations include the nature of the variables (discrete, continuous, binary), scales of measurement (nominal, ordinal, interval, ratio), and specific knowledge about the subject matter.

We now detail the procedure used for performing a k-nearest neighbours in the user-based indirect method (see Section 7.3). The procedure used for performing a k-nearest neighbours in the movie-based indirect method is similar and does not require further explanations. In the case of our movie database, pairs of items are compared on the basis of the presence or absence of certain features. The presence or absence of a feature is described mathematically by using a binary variable, which takes the value 1 if the feature is present (i.e., if person i has watched movie j) and the value 0 if the feature is absent (i.e., if person i has not watched movie j). More precisely, each person i is characterized by a binary vector $\mathbf{v}_i$ encoding the movies that that person watched. The dimension of this vector is equal to the number of movies. The k-nearest neighbours of person i are computed by taking the k nearest $\mathbf{v}_j$ according to a given similarity measure between binary vectors, $\text{sim}(i, j) = s(\mathbf{v}_i, \mathbf{v}_j)$.

We performed systematic comparisons between eight different such measures s (see Table 7.1 taken from [36]) and for different values of k ($= 1, 2, ..., 10, 20, ..., 100$). Table 7.1 lists these common similarity coefficients defined for binary features in terms of the frequencies in Table 7.2. In this table 7.2, a represents the frequency of 1-1 matches between $\mathbf{v}_i$ and $\mathbf{v}_j$, b is the frequency of 1-0 matches, and so forth. A short

Coefficient	Rationale
1. $\frac{a+d}{p}$	Equal weights for 1-1 matches and 0-0 matches.
2. $\frac{2(a+d)}{2(a+d)+b+c}$	Double weight for 1-1 matches and 0-0 matches.
3. $\frac{a+d}{a+d+2(b+c)}$	Double weight for unmatched pairs.
4. $\frac{a}{p}$	No 0-0 matches in numerator.
5. $\frac{a}{a+b+c}$	No 0-0 matches in numerator or denominator. (The 0-0 matches are treated as irrelevant).
6. $\frac{2a}{2a+b+c}$	No 0-0 matches in numerator or denominator. Double weight for 1-1 matches.
7. $\frac{a}{a+2(b+c)}$	No 0-0 matches in numerator or denominator. Double weight for unmatched pairs.
8. $\frac{a}{b+c}$	Ratio of matches to mismatches with 0-0 matches excluded.

Table 7.1: Similarity coefficients

rationale follows the definition of each measure in Table 7.1. The best scores (for all the performance measures) were obtained with the "a ratio of 1-1 matches to mismatches with 0-0 matches excluded". Notice that the k-nearest neighbours is an indirect method that cannot be used in the direct method.

In the sequel, we only present the results obtained by the best k-nearest neighbours model (i.e., $\text{sim}(i,j) = a/(b+c)$ and $k = 100$).

Cosine coefficient (Cosine). The cosine coefficient between persons i and j, which measures the strength and the direction of a linear relationship between two variables, is defined by $\text{sim}(i,j) = \text{cosine}(i,j) = (\mathbf{v}_i{}^{\text{T}}\mathbf{v}_j)/(\|\mathbf{v}_i\|\,\|\mathbf{v}_j\|)$. We again systematically varied the number k of neighbours ($= 1, 2, ..., 10, 20, ..., 100$). The

		Individual j		
		1	0	Totals
Individual i	1	a	b	$a+b$
	0	c	d	$c+d$
Totals		$a+c$	$b+d$	$p = a+b+c+d$

Table 7.2: Contingency table.

cosine coefficient algorithm is also an indirect method that cannot be used in the direct method.

Katz. This similarity index has been proposed in the social sciences field and has been recently rediscovered in the context of collaborative recommendation [33] and kernel methods where it is known as the von Neumann kernel [57]. Katz proposed in [38] a method of computing similarities, taking into account not only the number of direct links between items but, also, the number of indirect links (going through intermediaries) between items. The similarity matrix is

$$\mathbf{T} = \alpha\mathbf{A} + \alpha^2\mathbf{A}^2 + ... + \alpha^k\mathbf{A}^k + ... = (\mathbf{I} - \alpha\mathbf{A})^{-1} - \mathbf{I} \tag{7.3}$$

where $\mathbf{A}$ is the adjacency matrix and α is a constant which has the force of a probability of effectiveness of a single link ($\text{sim}(i,j) = t_{ij} = [\mathbf{T}]_{ij}$). A k-step chain or path, then, has a probability α^k of being effective. In this sense, α actually measures the attenuation in a link, $\alpha = 0$ corresponding to complete attenuation and $\alpha = 1$ to absence of any attenuation. For the series to be convergent, α must be less than the inverse of the spectral radius of $\mathbf{A}$.

For the experiment, we systematically varied the value of α ($\alpha = (0.05, 0.10, ..., 0.95) *$ (spectral radius)$^{-1}$) and we present only the results obtained by the best model (namely, $\alpha = 0.05 *$ (spectral radius)$^{-1}$).

Random-forest based algorithm (RFA). The similarity matrix introduced by Chebatorev & Shamis in [14], [16] is

$$\mathbf{T} = (\mathbf{I} + \mathbf{L})^{-1} \tag{7.4}$$

where $\mathbf{L}$ is the Laplacian matrix. This similarity measure has an interesting interpretation in terms of the matrix-forest theorem [14], [16]. Suppose that $F^i(G) = F^i$ is the set of all spanning forests rooted on node i of graph G, and $F^{ij}(G) = F^{ij}$ is the set of those spanning rooted forests for which nodes i and j belong to the same tree rooted at i. A spanning rooted forest is an acyclic subgraph of G that has the same nodes set as G and one marked node (a root) in each component. It is shown in [14], [16] that the matrix $(\mathbf{I}+\mathbf{L})^{-1}$ exists and that $[(\mathbf{I}+\mathbf{L})^{-1}]_{ij} = \epsilon(F^{ij})/\epsilon(F^i)$ where $\epsilon(F^{ij})$ and $\epsilon(F^i)$ are the total weights of forests that belong to F^{ij} and F^i respectively. The elements of this matrix are therefore called "relative forest accessibilities" between nodes. It can be shown that this matrix is a similarity measure, having the natural properties of a similarity (triangular property for similarities, among others [15]). It has recently been rediscovered and used in the context of documents ranking [35]. It is clear that this similarity measure is closely related to $\mathbf{L}^+$. Indeed, $(\mathbf{I}+\mathbf{L})^{-1}$ and $\mathbf{L}^+$ have the same set of eigenvectors, and their eigenvalues are related by $\lambda_i^{\text{RFA}} = \lambda_i^{\mathbf{L}^+}/(1+\lambda_i^{\mathbf{L}^+})$. This operation is similar to a regularization of the Laplacian matrix.

Shortest-path algorithm (Dijkstra). This algorithm solves a shortest path problem for a directed and connected graph with nonnegative edge weights. As a distance between two elements of the database, we compute the shortest path between these two elements. We do not show in the sequel the results of the shortest path algorithm. Indeed, it seems that, for example using the direct method, nearly each movie can be reached from any person with a shortest path distance of 3. The degree of agreement is therefore close to 50% because of the large number of ties, and the detailed results are of little interest.

8. Cross-validation results

8.1. Ranking procedure

Thus, for each step of the cross-validation and for each person, we first select the movies that have not been watched. Then, we rank them according to all the described scoring algorithms and the ways to use them (direct, user-based indirect or movie-based indirect method). Finally, we compare the proposed ranking with the `test set` (if the ranking procedure performs well, we expect watched movies belonging to the `test set` to be on top of the list) by using the three measures of performance.

8.2. Results

All the results are summarized in Table 9.1, which shows the three performance measures: the degree of agreement (Agreement), the percentile score (Percentile), and the recall, considering either the top 10 of the ranked list (Recall 10) or the top 20 of the ranked list (Recall 20). The standard deviation of the results (STD) across the 10 cross-validation sets is also reported, as well as the optimal number of neighbours (Neighbours), when applicable.

8.3. Discussion of the results

Table 9.1 shows that, when using the direct method to rank the movies for each user, the best results are obtained by $\mathbf{L}^+$, $\cos^+$, and RFA. Notice that we use a paired t-test to determine if there is a significant difference (with a p-value smaller than 10^{-2}) between the results of the various scoring algorithms. The best results, for each measure of performance and for each method (i.e., direct, user-based indirect, or movie-based indirect), are displayed in bold in each row of the table.

In the user-based indirect method, the bests results are obtained by $\mathbf{L}^+$ and RFA. In particular, the best degree of agreement and percentile are provided by $\mathbf{L}^+$ whereas the best recall scores (either the recall 10, or the recall 20) are obtained by both scoring algorithms with no significant difference. In the movie-based indirect method, kNN and RFA provide the best results.

When looking at the global performance (regardless of the direct or indirect way the similarities are computed) of the various scoring algorithms, Table 9.1 shows that $\mathbf{L}^+$, kNN, Cosine, $\cos^+$, and RFA are the best scoring algorithms in terms of both performance and stability of the results. The **best results overall** are obtained by the kNN, used in the movie-based indirect method (as proposed in the SUGGEST method, [37]), when considering the degree of agreement or the percentile, by $\mathbf{L}^+$ and RFA, used in the user-based indirect method, when considering the recall scores (considering either the top 10 or the top 20). We see that the user-based indirect method provides better recommendations than the movie-based indirect method for $\mathbf{L}^+$ and RFA, and for both kNN and Cosine but only for recall scores.

All the dissimilarity measures (i.e., CT, One-way and Return) are clearly less efficient (except the Return in the user-based indirect method) and seem to present a lack of stability (the results highly depends on the method used). We observe that CT and One-way give better results in the direct method than in the indirect method and that their direct recommendations are very similar to simply suggest the most popular movies (MaxF), most likely because they are both dominated by the stationary distribution, as already stated in [8]. Indeed, Brand [8] showed that the commute time is highly sensitive to the degree of the node (which is equal to the stationary distribution up to a scaling factor when considering a simple random walk on a graph). His work therefore confirms our results.

The fact that the inner products ($\mathbf{L}^+$, $\cos^+$, and RFA) provide better results than the corresponding distance measures (CT, PCA CT, One-way, and Return) shows that, in these experiments, the angle between the node vectors is a much more predictive measure than the distance between the nodes. The situation is therefore quite similar to what we observe in information retrieval. This result was also pointed out by Brand [8].

In conclusion, three similarity measures provide very good and stable performance: $\mathbf{L}^+$, RFA and the simple nearest neighbours technique (kNN). We also clearly observe that using the most efficient scoring algorithms ($\mathbf{L}^+$, RFA) in the indirect method improves the results, in comparison with the direct method. It is not clear, however, which of the two different indirect methods perform best. Indeed, kNN provides slightly better results in the movie-based indirect method, while we observe the opposite for $\mathbf{L}^+$ and RFA.

Correlations between the ranking algorithms. The Kendall rank-order correlation coefficient provides a measure of the degree of association or correlation between two sets of rankings (see [60] for details; its range is $[0, 1]$ with 1 corresponding to a perfect correlation and 0.5 corresponding to no association at all). Table 9.2 shows the average correlations between the rankings provided by all the ranking algorithms.

Firstly, we observe that the values of the correlations are quite low (rarely more than 0.75). This can be partially explained by the features of the database. There are actually many movies that have been watched by few people (2 or less). These movies have a small influence on the measure of performance (i.e., it is rare to find one of them in the movies belonging to the `test set` for a specific user) whereas they have some influence on the Kendall scores, because of their quantity (remember that a Kendall score is computed using the whole rankings provided by the considered scoring algorithms).

We observe that, as discovered in the previous section, MaxF is positively correlated with CT, PCA CT, and One-way used in the direct method. On the other hand, $\mathbf{L}^+$, $\cos^+$, Katz, and RFA are negatively correlated with MaxF. Thus, these methods do not always suggest best-sellers.

8.4. Computing times

In this section, we perform a comparison of the computing time (on a Pentium 4, 2.80 GHz) for all the implemented scoring algorithms and the ways defined to use them (direct, user-based indirect or movie-based indirect method). Notice that we do not use (for none scoring algorithm) the sparseness of the adjacency matrix $\mathbf{A}$ in order to reduce the computing times.

Table 8.1 shows the time, in seconds (using the Matlab `cputime` function), needed by each scoring algorithm to compute, from the adjacency matrix $\mathbf{A}$, a $n \times n$ matrix whose element i,j is a similarity measure between node i and node j. Notice that this matrix could be computed offline so that it could take very little time to provide an online recommendation.

All these scoring algorithms were implemented in Matlab. We used, in order to compute the similarity matrices, Equation 5.6 for the average commute time, the method suggested by Kemeny & Snell ([39], p.218) for the average first-passage times (one-way and return), Equation 5.1 for $\mathbf{L}^{+}$ and the derived $\cos^{+}$, and Equations 7.3 and 7.4 (both by inverting the matrix in Matlab) for respectively Katz and RFA similarity matrices. Notice also that, for the principal components analysis, we first had to compute the $\mathbf{L}^{+}$ matrix, then take out its eigenvectors and eigenvalues (using the Matlab `svd` function) and finally compute the derived distances.

MaxF	CT	PCA CT	One-way	Return	$\mathbf{L}^{+}$	$\cos^{+}$	kNN	Cosine	Katz	RFA
0	327	1308	630	630	25	228	114	202	25	25

Table 8.1: Time (in sec) needed to compute predictions for all the non-watched movies and all the users.

We observe that the slowest methods are the distance-based scoring algorithms (i.e., CT, PCA CT, One-way, and Return) and the $\cos^{+}$ method. The fastest scoring algorithms (if we do not consider the MaxF algorithm which provides nearly immediate results) are $\mathbf{L}^{+}$, Katz, and RFA.

9. Conclusions and further work

We introduced a general procedure for computing similarities between elements of a database. It is based on a Markov-chain model of random walk through the database. More precisely, we compute quantities (the average first-passage time, the average commute time, and the pseudoinverse of the Laplacian matrix) that provide similarity measures between any pair of elements. These similarity measures can be used in order to compare items belonging to database tables that are not necessarily directly connected. It relies on the degree of connectivity between these items.

Notice that, while the theoretical framework has been developed in the case of a weighted, undirected, graph, the notions of average first-passage time and average commute time also apply to **directed** (although strongly connected) graphs. Thus, these similarity measures can be used in the case of directed graphs as well. However, the nice interpretation in terms of the pseudoinverse of the Laplacian matrix does not apply in this case.

We showed through experiments performed on the MovieLens database that inner-product based quantities perform well in comparison with standard scoring algorithms. In fact, as already stressed by [40], the introduced quantities provide a very general mechanism for computing similarities between nodes of a graph, by exploiting its structure.

More precisely, the experiments showed that three similarity measures provide good and stable performance: the pseudoinverse of the Laplacian matrix, the random-forest based similarity measure, and the simple nearest neighbour technique (the later used in an item-based indirect method, such as proposed in the SUGGEST methodology [37]). However, for the nearest neighbour, two parameters need to be adjusted: the number of neighbours and the similarity between binary vectors. Moreover, the pseudoinverse of the Laplacian matrix, the random-forest based similarity measure are much more generic than the nearest neighbour since they provide similarity measures between any two elements of a database.

The main drawback of this method is that it does not scale well for large databases. Indeed, the Markov model has as many states as elements in the database. For large databases, we have to rely on iterative formula or approximate algorithms and on the sparseness of the matrix.

We are now working on other generalizations of standard multivariate statistical methods to the mining of databases, such as discriminant analysis. We are also comparing the different kernels on a graph that have been proposed in the literature on the same collaborative recommendation task.

References

[1] R. Baeza-Yates and B. Ribeiro-Neto. *Modern information retrieval*. Addison-Wesley, 1999.

[2] P. Baldi, P. Frasconi, and P. Smyth. *Modeling the Internet and the Web: Probabilistic Methods and Algorithms*. John Wiley and Sons, 2003.

[3] S. Barnett. *Matrices: Methods and Applications*. Oxford University Press, 1992.

[4] M. Belkin and P. Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems*, volume 14, pages 585–591. MIT Press, 2001.

[5] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15:1373–1396, 2003.

[6] A. Ben-Israel and T. Greville. *Generalized Inverses: Theory and Applications*. John Wiley and Sons, 1974.

[7] I. Borg and P. Groenen. *Modern multidimensional scaling: Theory and applications*. Springer, 1997.

[8] M. Brand. A random walks perspective on maximizing satisfaction and profit. *Technical Report TR2005-050, Mitsubishi Electric Research Laboratories*, 2005.

[9] P. Bremaud. *Markov Chains: Gibbs Fields, Monte Carlo Simulation, and Queues*. Springer-Verlag, 1999.

[10] F. Buckley and F. Harary. *Distance in graphs*. Addison-Wesley Publishing Company, 1990.

[11] S. Campbell and C. Meyer. *Generalized inverses of linear transformations*. Pitman Publishing Company, 1979.

[12] T. Chan, P. Ciarlet, and W. Szeto. On the optimality of the median cut spectral bisection graph partitioning method. *SIAM Journal on Scientific Computing*, 18(3):943–948, 1997.

[13] A. K. Chandra, P. Raghavan, W. L. Ruzzo, R. Smolensky, and P. Tiwari. The electrical resistance of a graph captures its commute and cover times. *Annual ACM Symposium on Theory of Computing*, pages 574–586, 1989.

[14] P. Chebatorev and E. Shamis. The matrix-forest theorem and measuring relations in small social groups. *Automation and Remote Control*, 58(9):1505–1514, 1997.

[15] P. Chebatorev and E. Shamis. On a duality between metrics and s-proximities. *Automation and Remote Control*, 59(4):608–612, 1998.

[16] P. Chebatorev and E. Shamis. On proximity measures for graph vertices. *Automation and Remote Control*, 59(10):1443–1459, 1998.

[17] S. Cheng and N. Higham. A modified cholesky algorithm based on a symmetric indefinite factorization. *SIAM Journal on Matrix Analysis and Applications*, 19(4):1097–1110, 1998.

[18] K. Cheung, K. Tsui, and J. Liu. Extended latent class models for collaborative recommendation. *IEEE Transactions on Systems, Man, and Cybernatics. Part A: Systems and Humans*, 34:143–148, 2004.

[19] F. R. Chung. *Spectral graph theory*. American Mathematical Society, 1997.

[20] C. Ding. Spectral clustering. *Tutorial presented at the 16th European Conference on Machine Learning (ECML 2005)*, 2005.

[21] P. G. Doyle and J. L. Snell. *Random Walks and Electric Networks.* The Mathematical Association of America, 1984.

[22] S. Dzeroski. Multi-relational data mining: an introduction. *ACM SIGKDD Explorations Newsletter*, 5(1):1–16, 2003.

[23] C. Faloutsos, K. McCurley, and A. Tomkins. Fast discovery of connection subgraphs. *Proceedings of the tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 118–127, 2004.

[24] M. Fiedler. A property of eigenvectors of nonnegative symmetric matrices and its applications to graph theory. *Czechoslovak Mathematical Journal*, 25(100):619–633, 1975.

[25] F. Fouss, A. Pirotte, J.-M. Renders, and M. Saerens. A novel way of computing similarities between nodes of a graph, with application to collaborative recommendation. *Proceedings of the 2005 IEEE/WIC/ACM International Joint Conference on Web Intelligence*, pages 550–556, 2005.

[26] F. Gobel and A. Jagers. Random walks on graphs. *Stochastic Processes and their Applications*, 2:311–336, 1974.

[27] F. A. Graybill. *Matrices with Applications in Statistics.* Wadsworth International Group, 1983.

[28] A. Greenbaum. *Iterative Methods for Solving Linear Systems.* Society for Industrial and Applied Mathematics, 1997.

[29] D. Harel and Y. Koren. On clustering using random walks. *Proceedings of the conference on the Foundations of Software Technology and Theoretical Computer Science; Lecture Notes in Computer Science*, 2245:18–41, 2001.

[30] J. Herlocker, J. Konstan, L. Terveen, and J. Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems*, 22(1):5–53, 2004.

[31] I. Herstein and D. Winter. *Matrix theory and linear algebra.* Maxwell Macmillan International Editions, 1988.

[32] N.-D. Ho and P. V. Dooren. On the pseudo-inverse of the laplacian of a bipartite graph. *Applied Mathematics Letters*, 18(8):917–922, 2005.

[33] Z. Huang, H. Chen, and D. Zeng. Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering. *ACM Transactions on Information Systems*, 22(1):116–142, 2004.

[34] D. Isaacson and R. Madsen. *Markov chains theory and applications.* John Wiley and Sons, 1976.

[35] T. Ito, M. Shimbo, T. Kudo, and Y. Matsumoto. Application of kernels to link analysis: First results. In *Proceedings of the second workshop on mining graphs, trees and sequences, ECML/PKDD, Pisa*, pages 13–24, 2004.

[36] R. Johnson and D. Wichern. *Applied Multivariate Statistical Analysis, 5th Ed.* Prentice Hall, 2002.

[37] G. Karypis. Evaluation of item-based top-n recommendation algorithms. *Proceedings of the tenth International Conference on Information and Knowledge Management*, pages 247–254, 2001.

[38] L. Katz. A new status index derived from sociometric analysis. *Psychmetrika*, 18(1):39–43, 1953.

[39] J. G. Kemeny and J. L. Snell. *Finite Markov Chains.* Springer-Verlag, 1976.

[40] D. J. Klein and M. Randic. Resistance distance. *Journal of Mathematical Chemistry*, 12:81–95, 1993.

[41] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.

[42] R. I. Kondor and J. Lafferty. Diffusion kernels on graphs and other discrete structures. *Proceedings of the 19th International Conference on Machine Learning*, pages 315–322, 2002.

[43] B. Mohar. Laplace eigenvalues of graphs – a survey. *Discrete Mathematics*, 109:171–183, 1992.

[44] B. Nadler, S. Lafon, R. Coifman, and I. Kevrekidis. Diffusion maps, spectral clustering and eigenfunctions of fokker-planck operators. *Submitted for publication to Advances in Neural Information Processiong Systems*, 2005.

[45] M. Newman. A measure of betweenness centrality based on random walks. *Social Networks*, 27 (1):39–54, 2005.

[46] B. Noble and J. Daniels. *Applied linear algebra, 3th ed.* Prentice-Hall, 1988.

[47] J. Norris. *Markov Chains.* Cambridge University Press, 1997.

[48] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. *Technical Report, Computer System Laboratory, Stanford University*, 1998.

[49] C. Palmer and C. Faloutsos. Electricity based external similarity of categorical attributes. *Proceedings of the 7th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'03)*, pages 486–500, 2003.

[50] E. Parzen. *Stochastic Processes.* Holden-Day, 1962.

[51] A. Pothen, H. Simon, and K.-P. Liou. Partitioning sparse matrices with eigenvectors of graphs. *SIAM Journal on Matrix Analysis and Applications*, 11(3):430–452, 1990.

[52] C. Rao and S. Mitra. *Generalized inverse of matrices and its applications.* John Wiley and Sons, 1971.

[53] S. Ross. *Stochastic Processes, 2nd Ed.* Wiley, 1996.

[54] Y. Saad. *Iterative Methods for Sparse Linear Systems.* Society for Industrial and Applied Mathematics, 2000.

[55] M. Saerens, F. Fouss, L. Yen, and P. Dupont. The principal components analysis of a graph, and its relationships to spectral clustering. *Proceedings of the 15th European Conference on Machine Learning (ECML 2004). Lecture Notes in Artificial Intelligence, Vol. 3201, Springer-Verlag, Berlin*, pages 371–383, 2004.

[56] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Recommender systems for large-scale e-commerce: Scalable neighborhood formation using clustering. *Proceedings of the Fifth International Conference on Computer and Information Technology*, 2002.

[57] B. Scholkopf and A. Smola. *Learning with kernels.* The MIT Press, 2002.

[58] J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis.* Cambridge University Press, 2004.

[59] J. Shi and J. Malik. Normalised cuts and image segmentation. *IEEE Transactions on Pattern Matching and Machine Intelligence*, 22:888–905, August 2000.

[60] S. Siegel and J. Castellan. *Nonparametric Statistics for the Behavioral Sciences, 2nd Ed.* McGraw-Hill, 1988.

[61] A. J. Smola and R. Kondor. Kernels and regularization on graphs. In M. Warmuth and B. Schölkopf, editors, *Proceedings of the Conference on Learning Theory (COLT) and Kernels Workshop*, 2003.

[62] H. M. Taylor and S. Karlin. *An Introduction to Stochastic Modeling, 3th Ed.* Academic Press, 1998.

[63] S. Wasserman and K. Faust. *Social Network Analysis: Methods and Applications.* Cambridge University Press, 1994.

[64] S. White and P. Smyth. Algorithms for estimating relative importance in networks. *Proceedings of the ninth ACM SIGKDD International Conference on Knowledge Discovery and Data mining*, pages 266–275, 2003.

[65] D. Zhou, J. Huang, and B. Scholkopf. Learning from labeled and unlabeled data on a directed graph. *Proceedings of the 22nd International Conference on Machine Learning*, pages 1041–1048, 2005.

Acknowledgments
We thank P. Van Dooren and V. Blondel, from the "Département d'Ingénierie Mathématique" of the Université catholique de Louvain, for insightful discussions. Part of this work has been funded by a project with the Belgian Federal Police.

Direct method (in %)											
	MaxF	CT	PCA CT	One-way	Return	$\mathbf{L}^+$	$\cos^+$	kNN	Cosine	Katz	RFA
Agreement	85.98	85.98	86.90	85.96	80.11	91.11	90.52	/	/	88.38	**91.12**
STD	*0.32*	*0.33*	*0.32*	*0.33*	*0.32*	*0.17*	*0.24*	/	/	*0.30*	*0.17*
Percentile	10.73	10.73	10.04	10.74	17.88	**6.52**	7.37	/	/	8.93	**6.53**
STD	*0.45*	*0.45*	*0.54*	*0.45*	*0.44*	*0.30*	*0.39*	/	/	*0.41*	*0.30*
Recall 10	11.02	11.11	12.97	11.09	0.34	16.31	**17.24**	/	/	14.97	16.65
STD	*0.23*	*0.23*	*0.36*	*0.24*	*0.05*	*0.33*	*0.45*	/	/	*0.29*	*0.35*
Recall 20	17.43	17.57	21.77	17.54	1.07	26.39	26.16	/	/	23.11	**26.72**
STD	*0.43*	*0.43*	*0.67*	*0.44*	*0.16*	*0.48*	*0.50*	/	/	*0.41*	*0.53*
User-based indirect method (in %)											
Agreement	/	81.87	91.66	81.83	92.35	**92.90**	90.77	92.64	92.66	90.07	92.85
STD	/	*0.31*	*0.25*	*0.31*	*0.27*	*0.22*	*0.18*	*0.16*	*0.18*	*0.30*	*0.22*
Neighbours	/	100	100	100	100	100	40	100	70	30	100
Percentile	/	13.07	6.47	13.09	5.65	**5.19**	7.20	5.53	5.68	7.61	5.41
STD	/	*0.53*	*0.39*	*0.52*	*0.34*	*0.34*	*0.30*	*0.48*	*0.41*	*0.32*	*0.36*
Neighbours	/	100	60	100	100	60	30	50	30	30	60
Recall 10	/	10.93	18.57	10.91	20.56	**21.43**	17.18	20.76	20.68	17.07	**21.32**
STD	/	*0.35*	*0.39*	*0.35*	*0.23*	*0.41*	*0.31*	*0.33*	*0.22*	*0.48*	*0.38*
Neighbours	/	100	60	100	80	40	50	50	60	50	50
Recall 20	/	17.35	28.67	17.35	31.14	**32.20**	26.37	31.13	31.29	25.41	**32.28**
STD	/	*0.36*	*0.37*	*0.38*	*0.30*	*0.40*	*0.46*	*0.39*	*0.42*	*0.32*	*0.49*
Neighbours	/	100	60	100	60	50	40	50	40	30	40
Movie-based indirect method (in %)											
Agreement	/	61.28	90.74	85.76	83.25	92.13	89.52	**93.27**	92.78	86.80	92.21
STD	/	*1.06*	*0.26*	*0.51*	*0.24*	*0.22*	*0.16*	*0.22*	*0.17*	*0.26*	*0.17*
Neighbours	/	50	80	50	100	70	90	100	100	100	100
Percentile	/	35.02	7.13	10.91	14.64	5.76	8.72	**4.93**	5.26	10.41	5.42
STD	/	*2.78*	*0.40*	*0.42*	*0.74*	*0.24*	*0.26*	*0.32*	*0.26*	*0.40*	*0.24*
Neighbours	/	30	40	50	90	40	90	40	40	100	60
Recall 10	/	4.00	12.01	11.13	1.39	16.41	6.65	18.92	17.29	5.82	**19.84**
STD	/	*0.63*	*0.34*	*0.56*	*0.11*	*0.65*	*0.23*	*0.27*	*0.26*	*0.32*	*0.39*
Neighbours	/	40	20	100	20	40	30	20	20	100	60
Recall 20	/	6.93	20.80	17.45	3.88	27.98	12.59	**30.81**	28.72	10.77	**30.91**
STD	/	*0.80*	*0.34*	*0.62*	*0.22*	*0.50*	*0.30*	*0.38*	*0.44*	*0.48*	*0.39*
Neighbours	/	30	20	50	20	40	30	20	30	90	40

Table 9.1: Average results obtained by performing a 10-fold cross-validation by the various scoring algorithms, and the three methods defined to use them (direct, user-based indirect, and movie-based indirect).

Direct method										
	CT	PCA CT	One-way	Return	$\mathbf{L}^+$	$\cos^+$	kNN	Cosine	Katz	RFA
MaxF	0.7468	0.6303	0.7470	0.5248	0.2880	0.2861	/	/	0.2762	0.2870
CT		0.6298	0.9526	0.5251	0.2881	0.2861	/	/	0.2760	0.2871
PCA CT			0.6297	0.5369	0.3915	0.3873	/	/	0.3796	0.3906
One-way				0.5251	0.2882	0.2861	/	/	0.2761	0.2871
Return					0.4896	0.4827	/	/	0.4796	0.4890
$\mathbf{L}^+$						0.7222	/	/	0.7117	0.7309
$\cos^+$							/	/	0.7137	0.7172
kNN								/	/	/
Cosine									/	/
Katz										0.7127
User-based indirect method										
MaxF	0.2791	0.3263	0.2796	0.3865	0.3579	0.3263	0.3524	0.3502	0.2978	0.3557
CT		0.6695	0.7338	0.6075	0.6360	0.6681	0.6415	0.6442	0.6964	0.6378
PCA CT			0.6696	0.5903	0.6132	0.6494	0.6175	0.6193	0.6565	0.6272
One-way				0.6071	0.6359	0.6681	0.6412	0.6437	0.6961	0.6377
Return					0.5930	0.5988	0.5935	0.5941	0.6022	0.5930
$\mathbf{L}^+$						0.6223	0.6141	0.6148	0.6279	0.6327
$\cos^+$							0.6268	0.6302	0.6561	0.6239
kNN								0.6198	0.6326	0.6151
Cosine									0.6349	0.6159
Katz										0.6301
Movie-based indirect method										
MaxF	0.3520	0.4108	0.2749	0.4979	0.4109	0.3435	0.3837	0.3772	0.4570	0.4207
CT		0.5585	0.6547	0.4990	0.5565	0.4238	0.5715	0.5792	0.5363	0.4600
PCA CT			0.5902	0.5065	0.5392	0.5255	0.5492	0.5546	0.5210	0.5143
One-way				0.5023	0.5872	0.6544	0.6124	0.6207	0.5461	0.5748
Return					0.5049	0.5165	0.5029	0.5052	0.5016	0.5048
$\mathbf{L}^+$						0.5699	0.5653	0.5652	0.5269	0.5514
$\cos^+$							0.5848	0.5972	0.5405	0.5621
kNN								0.5897	0.5284	0.5689
Cosine									0.5310	0.5666
Katz										0.5273

Table 9.2: Correlation matrix containing the Kendall score for each pair of scoring algorithms, for the three methods defined to use them (direct, user-based indirect, and movie-based indirect).

A. Appendix: The probability of absorption and the average first passage time/cost

A.1. The probability of absorption

Assume that our Markov chain has n states, and that the last m states are the absorbing states, that is, $S^a = \{n-m+1, n-m+2, \ldots, n\}$. We now compute the probability that a random walker reaches state $k \in S^a$ before any other absorbing state, so that we consider that the random walker is "absorbed" by state k before being absorbed by any other state $\in S^a$ and $\neq k$:

$$\begin{aligned} u(k|i) &= \mathrm{P}(\text{absorption in } k|s(0)=i) \\ &= \sum_{j=1}^{n} \mathrm{P}(\text{absorption in } k|s(1)=j)\ \mathrm{P}(s(1)=j|s(0)=i) \\ &= \sum_{j=1}^{n} \mathrm{P}(\text{absorption in } k|s(1)=j)\, p_{ij} \end{aligned}$$

where state j is reached after one step. Now, there are three different cases: either state $j \leq n-m$ is a nonabsorbing state ($j \notin S^a$), or it is an absorbing state different from k ($j \in S^a$, $j \neq k$), or it is k itself ($j = k \in S^a$). If j is a nonabsorbing state, $\mathrm{P}(\text{absorption in } k|s(1)=j) = u(k|j)$. If j is an absorbing state different from k, reaching j precludes reaching k in the future, so that $\mathrm{P}(\text{absorption in } k|s(1)=j) = 0$. Finally, if $j = k$, we have $\mathrm{P}(\text{absorption in } k|s(1)=j) = 1$ since we reached state k.

We therefore have

$$\begin{aligned} u(k|i) &= \sum_{j=1}^{n-m} p_{ij}\, u(k|j) + \sum_{j=1, j\neq k}^{n} p_{ij} \cdot 0 + p_{ik} \cdot 1 \\ &= p_{ik} + \sum_{j=1}^{n-m} p_{ij}\, u(k|j), \text{ for } i = 1, \ldots, m \\ &= p_{ik} + \sum_{j=1, j\neq k}^{n} p_{ij}\, u(k|j), \text{ for } i \notin S^a \text{ and } k \in S^a \end{aligned} \tag{A.1}$$

Of course, when $i = k$, we have $u(k|k) = 1$ while if $i \neq k \in S^a$, $u(k|i) = 0$.

Therefore,

$$\begin{cases} u(k|i) = p_{ik} + \sum_{j=1, j\neq k}^{n} p_{ij}\, u(k|j), \text{ for } i \notin S^a \text{ and } k \in S^a \\ u(k|k) = 1, \text{ for } k \in S^a \\ u(k|i) = 0, \text{ for } i, k \in S^a \text{ and } i \neq k \end{cases} \tag{A.2}$$

A.2. The average first passage time/cost

We now derive a recurrence relation for computing $o(k|i)$ by first-step analysis. The total cost, $\gamma(k|i)$, incurred during a walk starting from state i and entering for the first time state k is

$$\gamma(k|i) = \sum_{t=1}^{T_{ik}} c(s(t)|s(t-1)) \tag{A.3}$$

Now, if we transform state k into an absorbing state so that $\mathrm{P}(s(t+1) = i|s(t) = k) = \delta(k,i)$, where $\delta(k,i)$ is the delta of Kronecker, the problem remains unchanged (the process stops once state k has been reached). If we further assume $c(k|k) = 0$, $\gamma(k|i)$ in Equation A.3 can be rewritten as

$$\gamma(k|i) = \sum_{t=1}^{\infty} c(s(t)|s(t-1))$$

since once we have reached state k, we remain in k forever and $c(k|k) = 0$. Furthermore, we immediately observe that $\gamma(k|k) = 0$ and therefore $E[\gamma(k|k)|s(0) = i] = 0$.

Let us now compute the quantity of interest, $E[\gamma(k|i)|s(0) = i]$, for $k \neq i$ where $c(s(t)|s(t-1))$ is a shortcut for $c(s(t) = i_t|s(t-1) = i_{t-1})$:

$$\begin{aligned}
&E[\gamma(k|i)|s(0) = i] = E\left[\sum_{t=1}^{\infty} c(s(t)|s(t-1))\,|s(0) = i\right] \\
= &\sum_{i_1,i_2,\ldots} \mathrm{P}(s(1) = i_1, s(2) = i_2, \ldots |s(0) = i) \left[\sum_{t=1}^{\infty} c(s(t)|s(t-1))\right] \\
= &\sum_{i_1,i_2,\ldots} \mathrm{P}(s(1) = i_1, s(2) = i_2, \ldots |s(0) = i) \\
&\times \left[c(s(1)|s(0)) + \sum_{t=2}^{\infty} c[s(t)|s(t-1)]\right] \\
= &\sum_{i_1} \mathrm{P}(s(1) = i_1|s(0) = i) \left\{ \sum_{i_2,i_3,\ldots} \mathrm{P}(s(2) = i_2, s(3) = i_3, \ldots |s(1) = i_1) \right. \\
&\left. \times \left[c(s(1)|s(0)) + \sum_{t=2}^{\infty} c\left(s(t)|s(t-1)\right)\right]\right\} \\
= &\sum_{i_1} p_{ii_1}[c(i_1|i) + E[\gamma(k|i_1)|s(0) = i_1]], \text{ for } i \neq k \\
= &\sum_{i_1} p_{ii_1} c(i_1|i) + \sum_{i_1} p_{ii_1} E[\gamma(k|i_1)|s(0) = i_1], \text{ for } i \neq k
\end{aligned}$$

where we used the Markov property.

Therefore, since $o(k|k) = 0$,

$$\left\{\begin{array}{l} o(k|k) = 0 \\ o(k|i) = \displaystyle\sum_{j=1}^{n} p_{ij}\, c(j|i) + \sum_{j=1, j\neq k}^{n} p_{ij}\, o(k|j), \text{ for } i \neq k \\ \end{array}\right. \tag{A.4}$$

For $m(k|i)$, $c(j|i) = 1$ and we obtain

$$\begin{cases} m(k|k) = 0 \\ m(k|i) = 1 + \displaystyle\sum_{j=1, j\neq k}^{n} p_{ij}\, m(k|j), \text{ for } i \neq k \end{cases} \tag{A.5}$$

These equations can be used in order to iteratively compute the first-passage times [47] or first-passage costs. The meaning of these formulae is quite obvious: in order to go from state i to state k, one has to go to any adjacent state j and proceed from there.

B. Appendix: Computation of the basic electrical quantities

In this section, we show how the general solution of the electrical network equations can be computed in function of the Laplacian matrix, for a network with a single current source at node a and a current sink at node b. The developments are largely inspired by [40]. From Kirchoff's law, we have

$$\begin{cases} \displaystyle\sum_{j \in N(i)} i_{ij} = 0, \text{ for } i \neq a, b \\ \displaystyle\sum_{j \in N(a)} i_{aj} = I \\ \displaystyle\sum_{j \in N(b)} i_{bj} = -I \end{cases}$$

where $N(k)$ is the set of nodes directly connected to node k (the neighbours), and the current $i_{ij} = -i_{ji}$. I is the current flowing from source a to sink b; in other words,

$$\sum_{j \in N(k)} i_{kj} = I\delta(k-a) - I\delta(k-b) \text{ for any node } k \tag{B.1}$$

where δ is the delta of Kronecker.

For the potential v_i, we have

$$i_{ij} = c_{ij}(v_i - v_j) \tag{B.2}$$

where the c_{ij} are the conductances.

By replacing B.2 in B.1, we easily obtain

$$\begin{aligned} I\delta(k-a) - I\delta(k-b) &= \sum_{j \in N(k)} c_{kj}(v_k - v_j) \\ &= \sum_{j \in N(k)} c_{kj} v_k - \sum_{j \in N(k)} c_{kj} v_j \\ &= v_k \sum_{j \in N(k)} c_{kj} - \sum_{j \in N(k)} c_{kj} v_j \\ &= d_{kk} v_k - \sum_{j \in N(k)} a_{kj} v_j \end{aligned} \tag{B.3}$$

since we define $a_{ij} = c_{ij}$, and $\mathbf{D} = \text{diag}(a_{i.})$, with $a_{i.} = \sum_{j=1}^{n} a_{ij}$ and $d_{ij} = [\mathbf{D}]_{ij}$. If we define $\mathbf{e}_i$ as the unit column vector, $\mathbf{e}_i = [\underset{1}{0}, \ldots, \underset{i-1}{0}, \underset{i}{1}, \underset{i+1}{0}, \ldots, \underset{n}{0}]^{\mathrm{T}}$, we can rewrite (B.3) in matrix form:

$$\begin{aligned} I(\mathbf{e}_a - \mathbf{e}_b) &= (\mathbf{D} - \mathbf{A})\boldsymbol{v} \\ &= \mathbf{L}\boldsymbol{v} \end{aligned}$$

where $\boldsymbol{v}$ is the vector of the potentials of the network nodes. Since the null space of $\mathbf{L}$ is spanned by $\mathbf{e}$ (see Section 2.1), we immediately deduce that

$$\boldsymbol{v} = I\mathbf{L}^{+}(\mathbf{e}_a - \mathbf{e}_b) + \mu\mathbf{e} \tag{B.4}$$

where μ is a scalar.

The second term of (B.4) indicates that the potential is defined up to a constant term. The difference of potential between any two nodes i, j is therefore

$$\begin{aligned} v_i - v_j &= \mathbf{e}_i^{\mathrm{T}}\boldsymbol{v} - \mathbf{e}_j^{\mathrm{T}}\boldsymbol{v} \\ &= (\mathbf{e}_i - \mathbf{e}_j)^{\mathrm{T}}\boldsymbol{v} \\ &= I(\mathbf{e}_i - \mathbf{e}_j)^{\mathrm{T}}\mathbf{L}^{+}(\mathbf{e}_a - \mathbf{e}_b) + \lambda(\mathbf{e}_i - \mathbf{e}_j)^{\mathrm{T}}\mathbf{e} \\ &= I(\mathbf{e}_i - \mathbf{e}_j)^{\mathrm{T}}\mathbf{L}^{+}(\mathbf{e}_a - \mathbf{e}_b) \end{aligned} \tag{B.5}$$

This indicates how the potential can be computed from $\mathbf{L}^{+}$; the currents are then easily obtained from (B.2).

C. Appendix: Some useful properties of the pseudoinverse of the Laplacian matrix

In this appendix, we prove some useful properties of $\mathbf{L}$ and $\mathbf{L}^{+}$.

C.1. $\mathbf{L}^{+}$ is symmetric

Since $\mathbf{L}$ is symmetric and, for any matrix $\mathbf{M}$, $(\mathbf{M}^{\mathrm{T}})^{+} = (\mathbf{M}^{+})^{\mathrm{T}}$ (see [3]), we easily obtain $\mathbf{L}^{+} = (\mathbf{L}^{\mathrm{T}})^{+} = (\mathbf{L}^{+})^{\mathrm{T}}$. Therefore, $\mathbf{L}^{+}$ is symmetric.

C.2. L is an EP-matrix

An EP matrix $\mathbf{M}$ is a matrix which commutes with its pseudoinverse, i.e. $\mathbf{M}^{+}\mathbf{M} = \mathbf{M}\mathbf{M}^{+}$. Since $\mathbf{L}$ is real symmetric, it is automatically an EP-matrix (see [3], p.253). In particular, the following properties are worth mentioning:

1. If $(\lambda_i \neq 0, \mathbf{u}_i)$ are (eigenvalues, eigenvectors) of $\mathbf{L}$, then $(\lambda_i^{-1} \neq 0, \mathbf{u}_i)$ are corresponding (eigenvalues, eigenvectors) of $\mathbf{L}^{+}$. On the other hand, if $(\lambda_j = 0, \mathbf{u}_j)$ are (eigenvalues, eigenvectors) of $\mathbf{L}$, then they are also (eigenvalues, eigenvectors) of $\mathbf{L}^{+}$.

2. In particular, $\mathbf{L}^{+}$ has rank $n - 1$ and has the same null space as $\mathbf{L}$: $\mathbf{L}^{+}\mathbf{e} = 0$ ($\mathbf{e}$ is the eigenvector associated to $\lambda_n = 0$).

3. The previous property implies that $\mathbf{L}^+$ is doubly centered (the sum of its columns and the sum of its rows are both zero), just like $\mathbf{L}$ (see also [52], chapter 10, for a discussion of this topic).

Other properties of EP-matrices are described in [3] or [11].

C.3. $\mathbf{L}^+$ is positive semidefinite

Indeed, from the previous property, the eigenvalues of $\mathbf{L}$ and $\mathbf{L}^+$ have the same sign and $\mathbf{L}$ is positive semidefinite; therefore $\mathbf{L}^+$ is also positive semidefinite.

D. Appendix: Computation of the probability of absorption

In this appendix, we derive formulae allowing us to compute the probabilities of absorption in terms of the Laplacian pseudoinverse. Our developments are based on the electrical equivalent.

Suppose we transform state a and state b into absorbing states. Remember that, when starting from node k, the probability of absorption by state a before absorption by state b ($u(a|k)$) is equivalent to the potential at node k (v_k) when a unit voltage is applied at node a ($v_a = 1$), while the voltage at node b is set to 0 ($v_b = 0$). It is easier to compute the difference $(v_k - v_b)$.

Let us first compute $v_a - v_b$ from (B.5):

$$v_a - v_b = 1 = I(\mathbf{e}_a - \mathbf{e}_b)^{\mathrm{T}}\mathbf{L}^+(\mathbf{e}_a - \mathbf{e}_b)$$

Therefore, $I = \left[(\mathbf{e}_a - \mathbf{e}_b)^{\mathrm{T}}\mathbf{L}^+(\mathbf{e}_a - \mathbf{e}_b)\right]^{-1}$. We are now ready to compute $\mathrm{P}_{abs}(a|k)$ from (B.5)

$$\begin{aligned}
u(a|k) &= \mathrm{P}_{abs}(a|k) \\
&= v_k - v_b \\
&= \frac{(\mathbf{e}_k - \mathbf{e}_b)^{\mathrm{T}}\mathbf{L}^+(\mathbf{e}_a - \mathbf{e}_b)}{(\mathbf{e}_a - \mathbf{e}_b)^{\mathrm{T}}\mathbf{L}^+(\mathbf{e}_a - \mathbf{e}_b)} \\
&= \frac{\mathbf{e}_k^{\mathrm{T}}\mathbf{L}^+\mathbf{e}_a - \mathbf{e}_k^{\mathrm{T}}\mathbf{L}^+\mathbf{e}_b - \mathbf{e}_b^{\mathrm{T}}\ \mathbf{L}^+\mathbf{e}_a + \mathbf{e}_b^{\mathrm{T}}\mathbf{L}^+\mathbf{e}_b}{\mathbf{e}_a^{\mathrm{T}}\mathbf{L}^+\mathbf{e}_a - \mathbf{e}_a^{\mathrm{T}}\mathbf{L}^+\mathbf{e}_b - \mathbf{e}_b^{\mathrm{T}}\mathbf{L}^+\mathbf{e}_a + \mathbf{e}_b^{\mathrm{T}}\mathbf{L}^+\mathbf{e}_b} \\
&= \frac{l^+_{ka} - l^+_{kb} - l^+_{ab} + l^+_{bb}}{l^+_{aa} + l^+_{bb} - 2l^+_{ab}} \qquad \text{(D.1)}
\end{aligned}$$

In the same way,

$$\begin{aligned}
u(b|k) &= \mathrm{P}_{abs}(b|k) \\
&= 1 - \mathrm{P}_{abs}(a|k) \\
&= \frac{l^+_{kb} - l^+_{ka} - l^+_{ab} + l^+_{aa}}{l^+_{aa} + l^+_{bb} - 2l^+_{ab}} \qquad \text{(D.2)}
\end{aligned}$$

In matrix form, we obtain

$$u(a|k) = \frac{(\mathbf{e}_k - \mathbf{e}_b)^{\mathrm{T}}\mathbf{L}^{+}(\mathbf{e}_a - \mathbf{e}_b)}{(\mathbf{e}_a - \mathbf{e}_b)^{\mathrm{T}}\mathbf{L}^{+}(\mathbf{e}_a - \mathbf{e}_b)} \tag{D.3}$$

$$u(b|k) = \frac{(\mathbf{e}_k - \mathbf{e}_a)^{\mathrm{T}}\mathbf{L}^{+}(\mathbf{e}_b - \mathbf{e}_a)}{(\mathbf{e}_a - \mathbf{e}_b)^{\mathrm{T}}\mathbf{L}^{+}(\mathbf{e}_a - \mathbf{e}_b)} \tag{D.4}$$

If there are several absorbing states with $\upsilon = 0$, all these states can be merged into one single state by summing the weights to the merged states.

E. Appendix: Computation of the average first-passage time/cost in function of $\mathbf{L}^{+}$

Let us start from Equations (3.3)

$$o(k|i) = \sum_{j=1}^{n} p_{ij}\,[(c(j|i) + o(k|j))]\,, \text{ for } i \neq k = \sum_{j=1}^{n} p_{ij} c(j|i) + \sum_{j=1}^{n} p_{ij} o(k|j), \text{ for } i \neq k \tag{E.1}$$

$$= r_i + \sum_{j=1, j\neq k}^{n} p_{ij} o(k|j), \text{ for } i \neq k \tag{E.2}$$

where $r_i = \sum_{j=1}^{n} p_{ij} c(j|i)$ for $i = 1 \ldots n$, and $o(k|k) = 0$ for all k. The corresponding $n \times 1$ column vector made of r_i is $\mathbf{r}$.

Let us renumber the states so that state k becomes state n, the last state of the Markov model, and rewrite Equation E.2 in matrix form. Notice, however, that the equation will refer to vectors and matrices where the nth row and the nth column have been canceled; we denote by $\widehat{\mathbf{o}}$, $\widehat{\mathbf{r}}$ and $\widehat{\mathbf{P}}$ the vectors/matrices obtained from $\mathbf{o}$, $\mathbf{r}$ and $\mathbf{P}$ by suppressing their nth row and column: $\widehat{\mathbf{o}} = \widehat{\mathbf{r}} + \widehat{\mathbf{P}}\widehat{\mathbf{o}} = \widehat{\mathbf{r}} + \widehat{\mathbf{D}}^{-1}\widehat{\mathbf{A}}\widehat{\mathbf{o}}$, where the column vector $\widehat{\mathbf{o}}$ has elements $[\widehat{\mathbf{o}}]_i = o(n|i)$ and $\widehat{\mathbf{P}} = \widehat{\mathbf{D}}^{-1}\widehat{\mathbf{A}}$. As long as there is no isolated node (with no edge associated to it), $\widehat{\mathbf{D}}$ can be inverted. If we pre-multiply this last equation by $\widehat{\mathbf{D}}$, we obtain $\widehat{\mathbf{D}}\widehat{\mathbf{o}} = \widehat{\mathbf{D}}\widehat{\mathbf{r}} + \widehat{\mathbf{A}}\widehat{\mathbf{o}}$; therefore $(\widehat{\mathbf{D}} - \widehat{\mathbf{A}})\widehat{\mathbf{o}} = \widehat{\mathbf{D}}\widehat{\mathbf{r}}$. By defining $\mathbf{b} = \mathbf{D}\mathbf{r}$, we finally obtain $\widehat{\mathbf{L}}\widehat{\mathbf{o}} = \widehat{\mathbf{b}}$ and, since $\mathbf{L}$ has rank $n-1$, $\widehat{\mathbf{L}}$ is of full rank. We therefore have $\widehat{\mathbf{L}}^{+} = \widehat{\mathbf{L}}^{-1}$, so that

$$\widehat{\mathbf{o}} = \widehat{\mathbf{L}}^{-1}\widehat{\mathbf{b}} = \widehat{\mathbf{L}}^{+}\widehat{\mathbf{b}} \tag{E.3}$$

or $o(n|i) = \sum_{j=1}^{n-1} \widehat{l}_{ij}^{-1} b_j$, for $i \neq n$.

We could solve these equations for all the nodes being node n in turn, but this would be quite inefficient. Instead, we will express the elements of $\widehat{\mathbf{L}}^{+}$ in terms of the elements of $\mathbf{L}^{+}$ in order to obtain a more general equation (valid for all the nodes). This can easily be done thanks to the general formula computing the pseudoinverse of a general $m \times n$ matrix $\mathbf{M}_n = [\mathbf{M}_{n-1}\ \mathbf{a}]$ obtained by appending a $m \times 1$ column vector $\mathbf{a}$ to the $m \times (n-1)$ matrix $\mathbf{M}_{n-1}$ (see, e.g., [3], [6]).

Before going into the details, here is a sketch of the main idea. From the $(n-1) \times (n-1)$ matrix $\widehat{\mathbf{L}}^{-1}$, we construct $\mathbf{L}^{+}$ by first adding a column vector to $\widehat{\mathbf{L}}$ and then a row vector, in order to obtain $\mathbf{L}$ and its corresponding pseudoinverse $\mathbf{L}^{+}$. Consequently we first obtain from $\widehat{\mathbf{L}}^{-1}$ a new $(n-1) \times n$ matrix, $\mathbf{M}_n = [\widehat{\mathbf{L}}\ \mathbf{a}]$, and compute its pseudoinverse $\mathbf{M}_n^+$ Then we add a row vector $\mathbf{a}'^{\mathrm{T}}$ to $\mathbf{M}_n$ in order to finally obtain $\mathbf{L} = \begin{bmatrix} \mathbf{M}_n \\ \mathbf{a}'^{\mathrm{T}} \end{bmatrix}$ and its pseudoinverse.

Here is the general formula (see [3] or [6]) allowing to compute the pseudoinverse of a matrix $\mathbf{M}_n = [\ \mathbf{M}_{n-1}\ \mathbf{a}]$ in terms of the pseudoinverse of $\mathbf{M}_{n-1}$:

$$\mathbf{M}_n^+ = \begin{bmatrix} \mathbf{M}_{n-1} & \mathbf{a} \end{bmatrix}^+ = \begin{bmatrix} \mathbf{M}_{n-1}^+ - \mathbf{d}\mathbf{b}^{\mathrm{T}} \\ \mathbf{b}^{\mathrm{T}} \end{bmatrix} \tag{E.4}$$

where we define $\mathbf{d} = \mathbf{M}_{n-1}^+\mathbf{a}$, $\mathbf{c} = \mathbf{a} - \mathbf{M}_{n-1}\mathbf{d}$, and

$$\mathbf{b}^{\mathrm{T}} = \begin{cases} \mathbf{c}^+ & \text{if } \mathbf{c} \neq \mathbf{0} \\ (1 + \mathbf{d}^{\mathrm{T}}\mathbf{d})^{-1}\mathbf{d}^{\mathrm{T}}\mathbf{M}_{n-1}^+ & \text{if } \mathbf{c} = \mathbf{0} \end{cases} \tag{E.5}$$

with $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$ and $\mathbf{d}$ being column vectors.

In our special case, $\mathbf{M}_{n-1} = \widehat{\mathbf{L}}$ is $(n-1)\times(n-1)$, and since we must obtain $\mathbf{L}\mathbf{e} = \mathbf{0}$ ($\mathbf{L}$ is doubly centered), the appended column $\mathbf{a}$ is minus the sum of the columns of $\widehat{\mathbf{L}}$; that is, $\mathbf{a} = -\widehat{\mathbf{L}}\widehat{\mathbf{e}}$. We thus have $\mathbf{d} = \widehat{\mathbf{L}}^+\mathbf{a} = -\widehat{\mathbf{L}}^{-1}\widehat{\mathbf{L}}\widehat{\mathbf{e}} = -\widehat{\mathbf{e}}$.

We will show that we do not need to explicitly compute $\mathbf{b}$; indeed, we obtain from Equation (E.4)

$$\mathbf{M}_n^+ = \begin{bmatrix} \widehat{\mathbf{L}}^+ + \widehat{\mathbf{e}}\mathbf{b}^{\mathrm{T}} \\ \mathbf{b}^{\mathrm{T}} \end{bmatrix} \tag{E.6}$$

By looking carefully at (E.6), we observe that $\widehat{\mathbf{L}}^+$ can be obtained from $\mathbf{M}_n^+$ by subtracting the nth row of $\mathbf{M}_n^+$ (that is, $\mathbf{b}^{\mathrm{T}}$) from all the rows of $\mathbf{M}_n^+$. Indeed, $\widehat{\mathbf{e}}\mathbf{b}^{\mathrm{T}}$ is a matrix repeating $\mathbf{b}^{\mathrm{T}}$ on all its rows:

$$\widehat{\mathbf{e}}\mathbf{b}^{\mathrm{T}} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \mathbf{b}^{\mathrm{T}} = \begin{bmatrix} \mathbf{b}^{\mathrm{T}} \\ \mathbf{b}^{\mathrm{T}} \\ \vdots \\ \mathbf{b}^{\mathrm{T}} \end{bmatrix}$$

In other words,

$$\mathbf{M}_n^+ = \begin{bmatrix} \widehat{\mathbf{L}}^+ \\ \mathbf{0}^{\mathrm{T}} \end{bmatrix} + \begin{bmatrix} \mathbf{b}^{\mathrm{T}} \\ \mathbf{b}^{\mathrm{T}} \\ \vdots \\ \mathbf{b}^{\mathrm{T}} \end{bmatrix}$$

This means that $[\widehat{\mathbf{L}}^+]_{ij} = \widehat{l}_{ij}^+ = [\mathbf{M}_n^+]_{ij} - [\mathbf{M}_n^+]_{nj}$.

Exactly the same reasoning holds when we append a $1 \times n$ row vector $\mathbf{a}'^{\mathrm{T}}$ to $\mathbf{M}_n$ in order to obtain $\mathbf{L} = \begin{bmatrix} \mathbf{M}_n \\ \mathbf{a}'^{\mathrm{T}} \end{bmatrix}$. It suffices to transpose $\mathbf{M}_n$ and add a column which verifies $\mathbf{a}' = -\mathbf{M}_n^{\mathrm{T}}\widehat{\mathbf{e}}$. Hence, $\mathbf{d} = (\mathbf{M}_n^{\mathrm{T}})^+\mathbf{a} = -(\mathbf{M}_n^{\mathrm{T}})^+\mathbf{M}_n^{\mathrm{T}}\widehat{\mathbf{e}} = -\widehat{\mathbf{e}}$ since the $n \times (n-1)$ matrix $\mathbf{M}_n^{\mathrm{T}}$ has rank $n-1$ and therefore $(\mathbf{M}_n^{\mathrm{T}})^+\mathbf{M}_n^{\mathrm{T}} = \mathbf{I}$ (this can be shown easily by computing the SVD form of the pseudoinverse matrix (see Theorem 6.2.16 of [27])). By using the same trick as before, we obtain $[\mathbf{M}_n^+]_{ij} = [\mathbf{L}^+]_{ij} - [\mathbf{L}^+]_{in}$.

This allows us to express the elements of $\widehat{\mathbf{L}}^+$ in terms of those of $\mathbf{L}^+$:

$$\widehat{l}_{ij}^+ = [\mathbf{M}_n^+]_{ij} - [\mathbf{M}_n^+]_{nj} = [\mathbf{L}^+]_{ij} - [\mathbf{L}^+]_{in} - [\mathbf{L}^+]_{nj} + [\mathbf{L}^+]_{nn} = l_{ij}^+ - l_{in}^+ - l_{nj}^+ + l_{nn}^+ \tag{E.7}$$

By substituting (E.7) in (E.3), we obtain

$$o(n|i) = \sum_{j=1}^{n-1} \widehat{l}_{ij}^{-1} b_j = \sum_{j=1}^{n-1} \left(l_{ij}^+ - l_{in}^+ - l_{nj}^+ + l_{nn}^+\right) b_j = \sum_{j=1}^{n} \left(l_{ij}^+ - l_{in}^+ - l_{nj}^+ + l_{nn}^+\right) b_j$$

since $\mathbf{L}^+$ is symmetric.

Now, remember that we reordered the numbering of the states in order to put state k at the nth row/column. For any arbitrary state, we thus have

$$o(k|i) = \sum_{j=1}^{n} \left(l_{ij}^+ - l_{ik}^+ - l_{kj}^+ + l_{kk}^+\right) b_j \tag{E.8}$$

where $b_i = \sum_{j=1}^{n} a_{ij} c(j|i)$. Notice that in the case of the average first-passage time, $c(j|i) = 1$, and $b_i = \sum_{j=1}^{n} a_{ij} c(j|i) = a_{i.} = d_{ii}$, the sum of the weights reaching node i. Therefore,

$$m(k|i) = \sum_{j=1}^{n} \left(l_{ij}^+ - l_{ik}^+ - l_{kj}^+ + l_{kk}^+\right) d_{jj} \tag{E.9}$$

F. Appendix: Computation of the average commute time in function of $\mathbf{L}^+$

Since we already have the formula for the average first-passage time (Equation E.9), computing the average commute time is trivial:

$$\begin{aligned} n(i,j) &= m(j|i) + m(i|j) = \sum_{k=1}^{n} \left(l_{ik}^+ - l_{ij}^+ - l_{jk}^+ + l_{jj}^+\right) d_{kk} + \sum_{k=1}^{n} \left(l_{jk}^+ - l_{ji}^+ - l_{ik}^+ + l_{ii}^+\right) d_{kk} \\ &= \sum_{k=1}^{n} \left(l_{ii}^+ + l_{jj}^+ - 2l_{ij}^+\right) d_{kk} = \left(l_{ii}^+ + l_{jj}^+ - 2l_{ij}^+\right) \sum_{k=1}^{n} d_{kk} = V_G \left(l_{ii}^+ + l_{jj}^+ - 2l_{ij}^+\right) \end{aligned} \tag{F.1}$$

Notice that Equation (F.1) can be put in matrix form:

$$n(i,j) = V_G \, (\mathbf{e}_i - \mathbf{e}_j)^{\mathrm{T}} \mathbf{L}^+ (\mathbf{e}_i - \mathbf{e}_j) \tag{F.2}$$

and we easily observe that $[n(i,j)]^{1/2}$ is a distance on the node space since $\mathbf{L}^+$ is positive semidefinite.

G. Appendix: Computation of the average commute time by means of electrical networks theory

We already know that, in the case of a single current source at a and a sink at b, the difference of potential between any two nodes i, j can be computed (B.5) as

$$v_i - v_j = I(\mathbf{e}_i - \mathbf{e}_j)^{\mathrm{T}} \mathbf{L}^+ (\mathbf{e}_a - \mathbf{e}_b)$$

The effective resistance is therefore the ratio of the difference of potential between nodes a and b on the current flowing from a to b:

$$\frac{v_a - v_b}{I} = (\mathbf{e}_a - \mathbf{e}_b)^{\mathrm{T}} \mathbf{L}^+ (\mathbf{e}_a - \mathbf{e}_b)$$

which is exactly what we found previously for the commute times, up to a proportionality constant, V_G.

H. Appendix: Mapping to a space preserving the ECTD

Let us show that the node vectors $\mathbf{e}_i$ can be mapped into a new Euclidean space that preserves the commute time distances. Indeed, every positive semidefinite matrix can be transformed to a diagonal matrix, $\mathbf{\Lambda} = \mathbf{U}^{\mathrm{T}}\mathbf{L}^{+}\mathbf{U}$, where $\mathbf{U}$ is an orthonormal matrix made of the eigenvectors of $\mathbf{L}^{+}$, $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_{n-1}, \mathbf{0}]$: the column vectors $\mathbf{u}_k$ are the orthonormal eigenvectors of $\mathbf{L}^{+}$, $\mathbf{u}_i^{\mathrm{T}}\mathbf{u}_j = \delta_{ij}$ or $\mathbf{U}^{\mathrm{T}}\mathbf{U} = \mathbf{I}$ (see for instance [46]). Hence we have

$$\begin{aligned} n(i,j) &= V_G\,(\mathbf{e}_i - \mathbf{e}_j)^{\mathrm{T}}\mathbf{L}^{+}(\mathbf{e}_i - \mathbf{e}_j) = V_G\,(\mathbf{x}_i - \mathbf{x}_j)^{\mathrm{T}}\mathbf{U}^{\mathrm{T}}\mathbf{L}^{+}\mathbf{U}(\mathbf{x}_i - \mathbf{x}_j) = V_G\,(\mathbf{x}_i - \mathbf{x}_j)^{\mathrm{T}}\mathbf{\Lambda}(\mathbf{x}_i - \mathbf{x}_j) \\ &= V_G\,(\mathbf{x}_i - \mathbf{x}_j)^{\mathrm{T}}\mathbf{\Lambda}^{1/2}\,\mathbf{\Lambda}^{1/2}(\mathbf{x}_i - \mathbf{x}_j) = V_G\,(\mathbf{x}_i - \mathbf{x}_j)^{\mathrm{T}}(\mathbf{\Lambda}^{1/2})^{\mathrm{T}}\mathbf{\Lambda}^{1/2}(\mathbf{x}_i - \mathbf{x}_j) = V_G\,(\mathbf{x}'_i - \mathbf{x}'_j)^{\mathrm{T}}(\mathbf{x}'_i - \mathbf{x}'_j) \end{aligned}$$

where we made the transformations

$$\begin{cases} \mathbf{e}_i = \mathbf{U}\mathbf{x}_i \text{ or } \mathbf{e}_i^{\mathrm{T}} = \mathbf{x}_i^{\mathrm{T}}\mathbf{U}^{\mathrm{T}} \\ \mathbf{x}'_i = \mathbf{\Lambda}_i^{1/2}\mathbf{x} \end{cases} \tag{H.1}$$

So, in this n-dimensional Euclidean space, the transformed node vectors, $\mathbf{x}'_i$, are exactly separated by Euclidean commute time distances.

Now, we easily observe from (H.1) that if u_i^k is coordinate i of eigenvector $\mathbf{u}_k$ ($[\mathbf{u}_k]_i = u_i^k$) corresponding to eigenvalue λ_k of $\mathbf{L}^{+}$, and if x_k^i is coordinate k of vector $\mathbf{x}_i$ ($[\mathbf{x}_i]_k = x_k^i$), we obtain $x_k^i = u_i^k$. We thus have $x_k'^i = \sqrt{\lambda_k}u_i^k$ where $x_k'^i$ is coordinate k of vector $\mathbf{x}'_i$ ($[\mathbf{x}'_i]_k = x_k'^i$).

In other words, the first coordinate of the n node vectors, $\mathbf{x}'_i, i = 1 \ldots n$, corresponding to the **first axis** ($k = 1$) of the transformed space, are $x_1'^1, x_1'^2, \ldots, x_1'^n$, or $\sqrt{\lambda_1}u_1^1, \sqrt{\lambda_1}u_2^1, \ldots, \sqrt{\lambda_1}u_n^1$. Thus, the first coordinate of these n node vectors is simply the projection of the original node vectors, $\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_n$, on the first eigenvector, $\mathbf{u}_1$, weighted by $\sqrt{\lambda_1}$. More generally, coordinate k of the node vectors in the transformed space is simply the projection of the original node vectors, $\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_n$, on $\mathbf{u}_k$, weighted by $\sqrt{\lambda_k}$. The idea developed in Section 6 is thus to discard the axes (eigenvectors) corresponding to the smallest eigenvalues of $\mathbf{L}^{+}$.

I. Appendix: $\mathbf{L}^{+}$ is a kernel

In this section, we show that $\mathbf{L}^{+}$ is a valid kernel a Gram matrix, see for instance [57]). Indeed, $\mathbf{L}^{+}$ is the matrix containing the inner products of the transformed vectors $\mathbf{x}'_i$:

$$\mathbf{x}_i'^{\mathrm{T}}\mathbf{x}'_j = (\mathbf{\Lambda}_i^{1/2}\mathbf{x}_i)^{\mathrm{T}}\mathbf{\Lambda}_j^{1/2}\mathbf{x}_j = \mathbf{x}_i^{\mathrm{T}}\mathbf{\Lambda}\mathbf{x}_j = \mathbf{e}_i^{\mathrm{T}}\mathbf{U}\mathbf{\Lambda}\mathbf{U}^{\mathrm{T}}\mathbf{e}_j = \mathbf{e}_i^{\mathrm{T}}\mathbf{L}^{+}\mathbf{e}_j = l_{ij}^{+}$$

Let us also show that the vectors $\mathbf{x}'_i$ are centered (their center of gravity is $\mathbf{0}$):

$$\sum_{i=1}^{n}\mathbf{x}'_i = \mathbf{\Lambda}^{1/2}\sum_{i=1}^{n}\mathbf{x}_i = \mathbf{\Lambda}^{1/2}\mathbf{U}^{\mathrm{T}}\sum_{i=1}^{n}\mathbf{e}_i = \mathbf{\Lambda}^{1/2}\mathbf{U}^{\mathrm{T}}\mathbf{e} \tag{I.1}$$

The columns of the orthonormal matrix $\mathbf{U}$ are made of the eigenvectors of $\mathbf{L}^{+}$ in decreasing order of corresponding eigenvalue. Since the smallest eigenvalue is $\lambda_n = 0$, and the corresponding (unnormalized) eigenvector is $\mathbf{e}$, all the columns of $\mathbf{U}$ are orthogonal to $\mathbf{e}$, except the nth one. Therefore, all the elements of the column vector

$\mathbf{U}^{\mathrm{T}}\mathbf{e}$ are zero, except the last (nth) one. But since we further multiply $\mathbf{U}^{\mathrm{T}}\mathbf{e}$ by the diagonal matrix $\mathbf{\Lambda}^{1/2}$ in (I.1), and the last diagonal element of $\mathbf{\Lambda}$ is 0 (remember that $\lambda_n = 0$), the last (n th) element of $\sum_{i=1}^{n} \mathbf{x}'_i$ is also equal to zero. We therefore obtain $\sum_{i=1}^{n} \mathbf{x}'_i = \mathbf{0}$.

Another way of proving the same fact would be to notice that, from $\mathbf{\Lambda} = \mathbf{U}^{\mathrm{T}}\mathbf{L}^{+}\mathbf{U}$, we have $\mathbf{\Lambda}^{1/2}\mathbf{U}^{\mathrm{T}} = \mathbf{\Lambda}^{-1/2}\mathbf{U}^{\mathrm{T}}\mathbf{L}^{+}$. Therefore, $\sum_{i=1}^{n} \mathbf{x}'_i = (\mathbf{\Lambda}^{1/2}\mathbf{U}^{\mathrm{T}})\mathbf{e} = (\mathbf{\Lambda}^{-1/2}\mathbf{U}^{\mathrm{T}}\mathbf{L}^{+})\mathbf{e} = \mathbf{0}$ since $\mathbf{L}^{+}\mathbf{e} = \mathbf{0}$.

Thus, if $\mathbf{X}'$ denotes the data matrix containing the coordinates of the nodes in the transformed space on each row:

$$\mathbf{X}' = [\mathbf{x}'_1, \mathbf{x}'_2, ..., \mathbf{x}'_n]^{\mathrm{T}}, \qquad (I.2)$$

we have $\mathbf{L}^{+} = \mathbf{X}'(\mathbf{X}')^{\mathrm{T}}$ with elements $l^{+}_{ij} = \mathbf{x}'^{\mathrm{T}}_i \mathbf{x}'_j$.

J. Appendix: Error bound on the approximation

The error we commit by using the approximation

$$\widetilde{n}(i,j) = V_G\,(\mathbf{e}_i - \mathbf{e}_j)^{\mathrm{T}}\widetilde{\mathbf{L}}^{+}(\mathbf{e}_i - \mathbf{e}_j) \qquad (J.1)$$

where $\widetilde{\mathbf{L}}^{+}$ is

$$\widetilde{\mathbf{L}}^{+} = \sum_{k=1}^{m} \lambda_k\, \mathbf{u}_k \mathbf{u}_k^{\mathrm{T}} \qquad (J.2)$$

with $m < n-1$, is bounded by

$$\begin{aligned}
||n(i,j) - \widetilde{n}(i,j)|| &= V_G\,||(\mathbf{e}_i - \mathbf{e}_j)^{\mathrm{T}}[\mathbf{L}^{+} - \widetilde{\mathbf{L}}^{+}](\mathbf{e}_i - \mathbf{e}_j)|| \\
&= V_G\,||(\mathbf{e}_i - \mathbf{e}_j)^{\mathrm{T}}\left[\sum_{k=m+1}^{n-1} \lambda_k\, \mathbf{u}_k \mathbf{u}_k^{\mathrm{T}}\right](\mathbf{e}_i - \mathbf{e}_j)|| \\
&= V_G\,||\sum_{k=m+1}^{n-1} \lambda_k (\mathbf{e}_i - \mathbf{e}_j)^{\mathrm{T}}\left[\mathbf{u}_k \mathbf{u}_k^{\mathrm{T}}\right](\mathbf{e}_i - \mathbf{e}_j)|| \\
&\leq V_G \sum_{k=m+1}^{n-1} \lambda_k || \left[(\mathbf{e}_i - \mathbf{e}_j)^{\mathrm{T}}\mathbf{u}_k\right]\left[\mathbf{u}_k^{\mathrm{T}}(\mathbf{e}_i - \mathbf{e}_j)\right] || \\
&\leq V_G \sum_{k=m+1}^{n-1} \lambda_k\, || \left[(\mathbf{e}_i - \mathbf{e}_j)^{\mathrm{T}}\mathbf{u}_k\right] ||^2 \\
&\leq V_G \sum_{k=m+1}^{n-1} \lambda_k\, ||\mathbf{e}_i - \mathbf{e}_j||^2 ||\mathbf{u}_k||^2 \\
&\leq V_G \sum_{k=m+1}^{n-1} \lambda_k\, ||\mathbf{e}_i - \mathbf{e}_j||^2 \\
&\leq 2V_G \sum_{k=m+1}^{n-1} \lambda_k
\end{aligned}$$

where $||\mathbf{e}_i - \mathbf{e}_j||^2 = 2$ and we used the triangle inequality as well as the Cauchy-Schwartz inequality. This way, we can limit the expansion in order to reach a given accuracy.

K. Appendix: Links with principal components analysis

K.1. Links with PCA: A first proof

We will now show that this decomposition is similar to principal components analysis in the sense that the projection has maximal variance among all the possible candidate projections. If $\mathbf{X}'$ denotes the data matrix containing the coordinates of the nodes in the transformed space, $\mathbf{x}_i'^{\mathrm{T}}$, on each row (see (I.2), in Appendix I), we easily deduce from (H.1) that $\mathbf{X}' = \mathbf{U}\mathbf{\Lambda}^{1/2}$.

It is well-known that the principal components analysis of a data matrix $\mathbf{X}'$ yields, as kth principal component, the eigenvector, $\mathbf{v}_k$, of $(\mathbf{X}')^{\mathrm{T}}\mathbf{X}'$ (which is the variance-covariance matrix, since the $\mathbf{x}_i'$ are centered).

But $(\mathbf{X}')^{\mathrm{T}}\mathbf{X}' = (\mathbf{U}\mathbf{\Lambda}^{1/2})^{\mathrm{T}}\mathbf{U}\mathbf{\Lambda}^{1/2} = \mathbf{\Lambda}$. Since $\mathbf{\Lambda}$ is a diagonal matrix, we deduce that the $\mathbf{x}_i'$ *are already expressed in the principal components coordinate system* – the eigenvectors of ($\mathbf{X}')^{\mathrm{T}}\mathbf{X}'$ are the basis vectors of the transformed space. Thus, if $x_k'^i$ is coordinate k of vector $\mathbf{x}_i'$, it corresponds to the projection of node i on the kth principal component. The variance, in terms of ECTD, of the nodes cloud on each principal component k is therefore λ_k. An alternative proof of the same result is presented in the next section.

We thus conclude that this projection can be viewed as a principal components analysis in the Euclidean space where the nodes are exactly separated by ECTD. This decomposition therefore defines the projection of the node vectors that has maximal variance (in terms of the ECTD) among all the possible candidate projections. It can be shown that performing a multidimensional scaling on the ECTD gives exactly the same results as the principal components analysis.

K.2. Links with PCA: An alternative point of view

In principal components analysis, the kth unit vector, $\mathbf{v}_k$ (ordered by decreasing importance), on which we project the data is provided by the eigenvalue/eigenvector equation

$$(\mathbf{X}')^{\mathrm{T}}\mathbf{X}'\mathbf{v}_k{=}\lambda_k\mathbf{v}_k$$

By pre-multiplying the equation by $\mathbf{X}'$, we obtain

$$\mathbf{X}'(\mathbf{X}')^{\mathrm{T}}(\mathbf{X}'\mathbf{v}_k){=}\lambda_k(\mathbf{X}'\mathbf{v}_k)$$

Thus, each eigenvector $\mathbf{v}_k$ of $(\mathbf{X}')^{\mathrm{T}}\mathbf{X}'$ corresponds to an eigenvector $\mathbf{X}'\mathbf{v}_k$ of the matrix $\mathbf{X}'(\mathbf{X}')^{\mathrm{T}} = \mathbf{L}^+$, associated to the same eigenvalue λ_k. If we denote by $\mathbf{u}_k$ the corresponding *unit eigenvector* ($||\mathbf{u}_k|| = 1$) of $\mathbf{X}'(\mathbf{X}')^{\mathrm{T}}$, we must have $\mathbf{X}'\mathbf{v}_k = c\,\mathbf{u}_k$, where c is some constant. Since $\mathbf{u}_k$ is a unit vector, we have $1 = \mathbf{u}_k^{\mathrm{T}}\mathbf{u}_k = c^{-2}\mathbf{v}_k^{\mathrm{T}}(\mathbf{X}')^{\mathrm{T}}\mathbf{X}'\mathbf{v}_k = c^{-2}\lambda_k$; hence $c = \sqrt{\lambda_k}$ and therefore $\mathbf{X}'\mathbf{v}_k = \sqrt{\lambda_k}\mathbf{u}_k$.

Now, we easily observe that the vector $\mathbf{X}'\mathbf{v}_k$ precisely represents *the projection of the data on the k th principal component*, $\mathbf{v}_k$, and therefore contains the coordinate k of the data vectors, $\mathbf{x}_i'$, in the principal components coordinate system: $\widetilde{x}_k'^i = [\mathbf{X}'\mathbf{v}_k]_i$. Thus, $\widetilde{x}_k'^i = \sqrt{\lambda_k}u_i^k$ with $u_i^k = [\mathbf{u}_k]_i$. But this corresponds exactly to the definition of the transformation 6.3; hence we have $\mathbf{x}_i' = \widetilde{\mathbf{x}}_i'$. The $\mathbf{x}_i'$ are therefore expressed in the principal components coordinate system.