Random-Walk Computation of Similarities between Nodes of a Graph with Application to Collaborative Recommendation

François Fouss, Alain Pirotte, *Member*, *IEEE*, Jean-Michel Renders, and Marco Saerens, *Member*, *IEEE*

Abstract—This work presents a new perspective on characterizing the similarity between elements of a database or, more generally, nodes of a weighted and 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 matrix (its elements are inner products closely related to commute times). A principal component analysis (PCA) of the graph is introduced for computing the subspace projection of the node vectors in a manner that preserves as much variance as possible in terms of the Euclidean commute-time distance. 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, it could be applied to machine-learning and pattern-recognition tasks involving a relational database.

Index Terms—Graph analysis, graph and database mining, collaborative recommendation, graph kernels, spectral clustering, Fiedler vector, proximity measures, statistical relational learning.

1 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 emphasize Web pages that are hubs and authorities (see [43]; called the HITS algorithm) has been well received in the community (for a review, see [4]).

This work views a database as a collection of sets of elements (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 general for computing similarities between nodes of a **weighted** and **undirected graph**.

Computing similarities between pairs of elements allows us, for instance, to determine the item that is most relevant (or 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 (or tables), `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 us to cluster them into groups with similar interest about watched movies.
- Computing similarities between people and movies allows us to suggest movies to watch or not to watch.
- Computing similarities between people and movie categories allows us 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, e.g., [41]) 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, e.g., [27]), provides a

- *F. Fouss, A. Pirotte, and M. Saerens are with the ISYS Unit, IAG, Université catholique de Louvain, Place des Doyens 1, B-1348 Louvain-la-Neuve, Belgium.*
  E-mail: {francois.fouss, alain.pirotte, marco.saerens}@ucLouvain.be.
- *J.-M. Renders is with the Learning and Content Analysis Group, Xerox Research Center Europe, Chemin de Maupertuis 6, 38240 Meylan (Grenoble) France. E-mail: jean-michel.renders@xrce.xerox.com.*

Manuscript received 13 Jan. 2005; revised 27 Dec. 2005; accepted 9 Aug. 2006; published online 18 Jan. 2007.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-0023-0105.

distance measure between any pair of states. The fact that this quantity is indeed a distance on a graph was proved independently by Klein and Randic [42] and Gobel and Jagers [27].

We will see later that $[n(i,j)]^{1/2}$, which is also a distance between nodes, takes a remarkable form and is Euclidean; 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, is the **pseudoinverse of the Laplacian matrix** of the graph ($\mathbf{L}^+$). The elements of $\mathbf{L}^+$ are the inner products of the node vectors in a Euclidean space preserving the ECTD (i.e., a Euclidean space where the nodes are exactly separated by the ECTD). $\mathbf{L}^+$ therefore provides a similarity measure between nodes and it is a valid kernel (a Gram matrix, see, e.g., [60]).

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, that is, when communication is facilitated. In short, the more short paths connect two given elements, the more similar those elements are. 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 ones. With a few notable exceptions (see the related work below), while being interesting alternatives to the well-known "shortest-path" distance on a graph [11], those quantities have not yet been fully exploited in the context of collaborative recommendation or, more generally, in pattern recognition and machine learning. In mathematical chemistry, on the other hand, attempts were made to use the "commute-time" distance instead of the "shortest-path" distance [42].

This paper is an extended and improved follow-up to two earlier papers: [58], [25]. The former introduces our theoretical model while the latter presents preliminary experimental results obtained on the collaborative-recommendation task. The present paper contains proofs of the main results, a discussion of computational issues, and an account of substantially expanded experiments.

1.1 Some Related Work

Chebotarev and Shamis already proposed in [15] and [17] a similarity measure between nodes of a graph integrating indirect paths, based on the matrix-forest theorem. Interestingly, this quantity is also related to the Laplacian matrix of the graph. While the authors prove some nice properties about their similarity measure, no experiment investigating its effectiveness was performed. We will therefore investigate this matrix-forest-based measure, together with other quantities, in Section 6.

Some authors recently considered similarity measures based on random-walk models. For instance, Harel and Koren [31] 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 that their algorithm is able to cluster arbitrary nonconvex shapes. White and Smyth [66], 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 et al. [51] by capturing a concept of "relative centrality" of a given node with respect to some other node of interest. On the other hand, Kondor and Lafferty [44] as well as Smola and Kondor [64] defined a graph regularization model which is closely related to the graph PCA introduced in Section 5. This model could be used in order to compute similarities between nodes, just as any other graph kernel [61].

More recently, Newman [48] suggested a random-walk model to compute a "betweenness centrality" of a given node in a graph. This counts how often a node is traversed during a random walk between two other nodes. Then, this quantity is averaged over every pair of nodes, providing a general measure of betweenness [65] 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 flows. In addition, Palmer and Faloutsos [52] define a similarity function between categorical attributes, called "refined escape probability," based on random walks and electrical networks. They show that this quantity provides a reasonably good measure for clustering and classifying categorical attributes. In a recent paper [47], Nadler et al. proposed a similarity 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 we are currently investigating its performance [26].

Very recently, Brand [10] proposed 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 latter is quite sensible to the node degree. Their conclusions confirm our experimental results, as will be shown in Section 6.

Our approach based on a random-walk model on a graph is also closely related to spectral-clustering and spectral-embedding techniques (for a recent account, see [20]), as detailed in [58]. Random-walk models on a graph also proved useful in the context of learning from labeled and unlabeled data (see, e.g., [67]).

1.2 Main Contributions

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 and Randic [42], 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 review some of its properties.
2. We show that the Moore-Penrose pseudoinverse of the Laplacian matrix of the graph is a valid kernel (a Gram matrix; see, for instance, [60]). It therefore defines a kernel on a graph and can be interpreted as a similarity measure.

3. We show that the node space of the graph can be projected 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. We also show that the ECTD PCA can be viewed as a special regularized Laplacian kernel, as introduced by Smola and Kondor [64].
4. From an experimental point of view, we show that these quantities can be used in the context of collaborative recommendation [18], [32], [35]. 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 and 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.

We stress that 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 usefulness of Markov-based similarity measures to nontrivial database mining tasks. This approach fits quite naturally into the so-called "multirelational data mining" and the "statistical relational learning" frameworks (see, for instance, [22] and other papers in the same issue of the SIGKDD newsletter).

1.3 Outline of the Paper

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 4 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. Section 5 introduces a subspace projection of the nodes of the graph that maximizes the variance of the projected data and makes the link with the Fiedler vector. Section 6 specifies our experimental methodology and illustrates the concepts with experimental results obtained on the MovieLens database. Section 7 is the conclusion.

2 A Markov-Chain Model of Database Navigation

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 ($w_{ij} \geq 0$) 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.

Thus, people who watch the same kind of movies, and therefore have similar taste, will be connected by a comparatively larger 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.

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 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 $\boldsymbol{\pi}(t) = [\pi_1(t), \pi_2(t), \ldots, \pi_n(t)]^{\mathrm{T}}$ at time t once the initial distribution $\boldsymbol{\pi}^0$ is known. For more details on Markov chains, the reader is invited to consult standard textbooks (e.g., [41], [50]).

3 Average First-Passage Time/Cost and Average Commute Time

This section reviews two basic quantities that can be computed from the definition of the Markov chain, that is, from its transition probability matrix: 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 us to compute these quantities are just introduced without proof (see, e.g., [41] or [50] for a more formal treatment).

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 [50]. 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 $c(j|i)$ for any states i, j. Notice that $m(k|i)$ is a special case of $o(k|i)$ obtained when $c(j|i) = 1$ for all i, j.

The recurrence relations for computing $m(k|i)$ and $o(k|i)$ can easily be obtained by first-step analysis [41], [50], [57]:

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

$$\begin{cases} o(k|k) = 0 \\ o(k|i) = \sum_{j=1}^{n} p_{ij}\, c(j|i) + \sum_{j=1}^{n} p_{ij}\, o(k|j), \quad \text{for } i \neq k. \end{cases} \tag{2}$$

These formulae are 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, [36], [41], [53]), or by using the pseudoinverse of the Laplacian matrix of the graph, as shown in this paper (see Section 4).

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 [27], [42], 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 [14]. We will show in Section 4 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 nodes has the desirable property of decreasing when the number of paths connecting the two nodes increases and when the length of paths decreases (see [21] for a proof based on electrical networks theory). This is indeed an intuitively satisfying property of the effective resistance of the equivalent electrical network [14], [21]. 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.

4 Computation of the Basic Quantities with $\mathbf{L}^+$

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 (1) and (2), 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 and Randic [42] who proved, based on the electrical equivalence, that the effective resistance (equivalent to 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 (1), and by providing formulae for the average first-passage time and the average first-passage cost. We are currently investigating the relationships between these results and those obtained by using the group generalized inverse, as proposed by Meyer in [45].

4.1 The Pseudoinverse of the Laplacian Matrix

The symmetric Laplacian matrix $\mathbf{L}$ of the graph is defined in the usual manner, $\mathbf{L} = \mathbf{D} - \mathbf{A}$, where $\mathbf{D} = \mathbf{Diag}(a_{i.})$ with $d_{ii} = [\mathbf{D}]_{ii} = a_{i.} = \sum_{j=1}^{n} a_{ij}$, 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 1s (i.e., $\mathbf{e} = [1, 1, \ldots, 1]^{\mathrm{T}}$, where T denotes the matrix transpose) and $\mathbf{0}$ is a column vector made of 0s, $\mathbf{L}\mathbf{e} = \mathbf{0}$ and $\mathbf{e}^{\mathrm{T}}\mathbf{L} = \mathbf{0}^{\mathrm{T}}$ 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]).

The **Moore-Penrose pseudoinverse** of $\mathbf{L}$ (see, for instance, [5]) will be denoted by $\mathbf{L}^+$, with elements $l_{ij}^+ = [\mathbf{L}^+]_{ij}$. The concept of pseudoinverse generalizes the matrix inverse to matrices that are not of full rank or not square. 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 (in which case it provides the solution closest to the origin). A thorough treatment of matrix pseudoinverses and their applications can be found in [8].

Appendix A reviews some useful properties of $\mathbf{L}^+$, in particular, 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 $\mathbf{L}^+$; 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 [55], chapter 10):

$$\mathbf{L}^+ = \left(\mathbf{L} - \mathbf{e}\mathbf{e}^{\mathrm{T}}/n\right)^{-1} + \mathbf{e}\mathbf{e}^{\mathrm{T}}/n, \tag{3}$$

where n is the number of nodes.

4.2 The Average First-Passage Time/Cost and Average Commute Time

Appendix B shows that the average first-passage time and the average first-passage cost can be computed in terms of $\mathbf{L}^+$ from (1) and (2). A similar formula (see Appendix C) is derived for the average commute time, which is repeated here

$$n(i,j) = V_G\left(l_{ii}^+ + l_{jj}^+ - 2l_{ij}^+\right), \tag{4}$$

where V_G is the volume of the graph ($V_G = \sum_{k=1}^{n} d_{kk}$). This formula was also obtained by using the electrical equivalent of the average commute time (the effective resistance) in [42]; see also [57].

If we define $\mathbf{e}_i$ as the ith column of $\mathbf{I}$,

$$\mathbf{e}_i = [\underset{1}{0}, \ldots, \underset{i-1}{0}, \underset{i}{1}, \underset{i+1}{0}, \ldots, \underset{n}{0}]^{\mathrm{T}},$$

(4) takes the remarkable form

$$n(i,j) = V_G\,(\mathbf{e}_i - \mathbf{e}_j)^{\mathrm{T}}\mathbf{L}^+(\mathbf{e}_i - \mathbf{e}_j), \tag{5}$$

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

We observe that $[n(i,j)]^{1/2}$ is a **distance** in the Euclidean space spanned by the node vectors of the graph since $\mathbf{L}^+$ is positive semidefinite. It is therefore **Euclidean** [28] because, as will be shown later, the nodes can be embedded in an Euclidean space exactly preserving the distances. This distance will be called the **Euclidean Commute-Time Distance** (ECTD). This is nothing else than a **Mahalanobis distance** with a weighting matrix $\mathbf{L}^+$.

4.3 Computational Issues

As the number of nodes increases, the computations directly based on the pseudoinverse eventually become intractable; using iterative techniques, taking advantage of the sparseness of the transition-probability matrix [30], [56], is one alternative (based on (1) or (2)).

Another alternative that proved useful for computing $\mathbf{L}^+$ for large, sparse, graphs is based on a sparse Cholesky factorization of the Laplacian matrix. First, observe that the ith column of $\mathbf{L}^+$, $\mathbf{l}_i^+ = \mathbf{col}_i(\mathbf{L}^+)$, can be obtained by the following procedure (see [33, pp. 440-441]):

1. Compute the projection of the basis vector $\mathbf{e}_i$ on the column space of $\mathbf{L}$: $\mathbf{y}_i = \mathbf{proj}_{\mathbf{L}}(\mathbf{e}_i) = (\mathbf{I} - \mathbf{e}\mathbf{e}^{\mathrm{T}}/n)\mathbf{e}_i$.
2. Find a solution $\mathbf{l}_i^{*+}$ of the equation $\mathbf{L}\mathbf{l} = \mathbf{y}_i$.
3. Project the result, $\mathbf{l}_i^{*+}$, on the row space of $\mathbf{L}$, $\mathbf{l}_i^+ = \mathbf{proj}_{\mathbf{L}}(\mathbf{l}_i^{*+}) = (\mathbf{I} - \mathbf{e}\mathbf{e}^{\mathrm{T}}/n)\mathbf{l}_i^{*+}$ (since $\mathbf{L}$ is symmetric, its row space is equal to its column space).

It can be easily shown, based on the electrical equivalence, that $\mathbf{l}_i^+$ represents the centered (summing to zero) voltage at each node when 1) a unit current is injected in node i and 2) a current $1/n$ is removed from every node ($\mathbf{e}\mathbf{e}^{\mathrm{T}}$ is a square matrix full of 1s).

The crux is to find one solution of $\mathbf{L}\mathbf{l} = \mathbf{y}_i$ in step 2 by using the Cholesky factorization. We first observe that, since the columns of $\mathbf{L}$ are linearly dependent, one arbitrary element of $\mathbf{l}$, say the last one, can be set to zero: $l_n = 0$. The resulting set of n equations is redundant (the rows of $\mathbf{L}$ and $\mathbf{y}_i$ sum to zero), and we can delete one of these equations, say the last one, without change. The resulting system of equations is equivalent to $\widehat{\mathbf{L}}\widehat{\mathbf{l}} = \widehat{\mathbf{y}}_i$ where $\widehat{\mathbf{L}}$ is the Laplacian matrix from which the last column and the last row have been deleted. Similarly, $\widehat{\mathbf{l}}$ and $\widehat{\mathbf{y}}_i$ are obtained from the $\mathbf{l}$ and $\mathbf{y}_i$ vectors, respectively, by removing their last row. The $(n-1)\times(n-1)$ matrix $\widehat{\mathbf{L}}$, called the reduced Laplacian matrix, is full rank and positive definite. A Cholesky factorization, $\widehat{\mathbf{L}} = \mathbf{R}\mathbf{R}^{\mathrm{T}}$, is performed. If $\widehat{\mathbf{L}}$ 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 adjacency matrix $\mathbf{A}$ in order to obtain a "band" matrix. Once the factorization has been computed, one solution of $\widehat{\mathbf{L}}\widehat{\mathbf{l}} = \mathbf{R}\mathbf{R}^{\mathrm{T}}\widehat{\mathbf{l}} = \widehat{\mathbf{y}}_i$, call it $\widehat{\mathbf{l}}_i^{*+}$, can easily be obtained by back-substitution ($\mathbf{R}$ is lower-triangular and sparse). A solution to $\mathbf{L}\mathbf{l} = \mathbf{y}_i$ in step 2 is therefore $(\mathbf{l}_i^{*+})^{\mathrm{T}} = [\widehat{\mathbf{l}}_i^{*+}, 0]$.

This procedure allows to compute the columns of $\mathbf{L}^+$ "on demand." We were thus able to compute the elements of $\mathbf{L}^+$ for sparse graphs of about 150,000 nodes.

Still another viable approach, based on a truncated series expansion, is proposed by Brand in [10]. Finally, the special case of a bipartite graph (as the movie database) is developed in [34], where the authors propose an optimized method for computing the pseudoinverse of the Laplacian matrix in this situation.

5 Maximum Variance Subspace Projection of the Node Vectors

5.1 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 D):

$$n(i,j) = V_G\,(\mathbf{x}_i' - \mathbf{x}_j')^{\mathrm{T}}(\mathbf{x}_i' - \mathbf{x}_j') = V_G\,\|\mathbf{x}_i' - \mathbf{x}_j'\|^2,$$

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 eigenvalue λ_k) and $\mathbf{\Lambda} = \mathbf{Diag}(\lambda_k)$. 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.

Appendix E shows that the $\mathbf{x}_i'$ are centered and that the elements of the pseudoinverse of the Laplacian matrix are the **inner products** between these transformed node vectors, $l_{ij}^+ = \mathbf{x}_i'^{\mathrm{T}}\mathbf{x}_j'$. Therefore, $\mathbf{L}^+$ is a **kernel** matrix (a Gram matrix) and can be considered as a **similarity matrix** for 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 [61], the diffusion kernel [44], and the recently introduced regularized Laplacian kernel [37], [64]. 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 Section 5.2). This result is worth mentioning since, once a meaningful kernel has been defined on a graph, a number of interesting measures come almost for free (kernel PCA, etc.; see, for

instance, [60]). We are currently comparing various other well-defined kernels on a graph on the same collaborative recommendation task [26].

One key issue here is to assess which of the **distance-based measures** (for instance, the ECTD) or the **inner-product-based measures** (for instance, $\mathbf{L}^+$) perform best for collaborative recommendation. It is well known that, for the vector-space model of information retrieval, inner-product-based measures outperform Euclidean distances when computing proximities between documents [3]. In the present case, ECTD are Euclidean distances, while $\mathbf{L}^+$ contains the inner products of node vectors. In this framework, another measure of interest is the **cosine** of node vectors, which is defined as

$$\cos^+(i,j) = l^+_{ij}/\sqrt{l^+_{ii}l^+_{jj}}. \tag{6}$$

5.2 Subspace Projection of the Node Vectors (The Principal Component 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 spectral decomposition

$$\widetilde{\mathbf{L}}^+ = \sum_{k=1}^{m} \lambda_k \mathbf{u}_k \mathbf{u}_k^{\mathrm{T}}, \tag{7}$$

where the $\mathbf{u}_k$ are the eigenvectors of $\mathbf{L}^+$ and λ_k the corresponding eigenvalues (see Appendix D for details). A new transformation of the node vectors is therefore defined by $\widetilde{\mathbf{x}}_i = \widetilde{\mathbf{U}}^{\mathrm{T}}\mathbf{e}_i$ and $\widetilde{\mathbf{x}}'_i = \widetilde{\mathbf{\Lambda}}^{1/2}\widetilde{\mathbf{x}}_i$, where $\widetilde{\mathbf{U}} = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_m, \mathbf{0}, \ldots, \mathbf{0}]$ and

$$\widetilde{\mathbf{\Lambda}} = \mathbf{Diag}[\lambda_1, \lambda_2, \ldots, \lambda_m, 0, \ldots, 0].$$

The $\widetilde{\mathbf{x}}'_i$ are column vectors containing zeroes from position $m+1$ on: $\widetilde{\mathbf{x}}' = [\widetilde{x}'_1, \widetilde{x}'_2, \ldots, \widetilde{x}'_m, 0, \ldots, 0]^{\mathrm{T}}$. This subspace is thus an m-dimensional space where the ECTD are approximately preserved. A bound on this approximation can easily be derived, $\|n(i,j) - \widetilde{n}(i,j)\| \leq V_G \sum_{k=m+1}^{n-1} \lambda_k$.

This decomposition is similar to Principal Component 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 [58]; see also Appendix F). This is related, in a number of interesting ways, with both spectral clustering (see, e.g., [62], [20], and our work [58]), kernel PCA [60], and spectral embedding [6], [7].

Moreover, it is easy to show that performing a multidimensional scaling on the ECTD gives exactly the same results as the PCA. Indeed, classical multidimensional scaling [9] amounts to performing the spectral decomposition of the matrix given by $-(1/2)\mathbf{HNH}$, and to retaining the first m eigenvalues and eigenvectors, where $\mathbf{H}$ is the centering matrix ($\mathbf{H} = \mathbf{I} - \mathbf{e}\mathbf{e}^{\mathrm{T}}/n$) and $\mathbf{N}$ is the squared ECTD matrix ($[\mathbf{N}]_{ij} = n(i,j)$). It is not difficult to show that $-(1/2)\mathbf{HNH}$ is nothing else than $\mathbf{L}^+$ (times V_G), so that both formulations are equivalent [9].

As for PCA, we expect that the first few principal components contain most of the information about the basic structure of the graph and that the remaining components related to smaller eigenvalues represent noise. If this is true, appropriate results are obtained by keeping only a few components of the PCA. For example, keeping only two or three components allows us to visualize the graph.

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

This result shows that the $\widetilde{\mathbf{L}}^+$ similarity matrix is a regularized Laplacian kernel (as defined in [44]) with a regularization factor given by $r(\lambda_i) = \lambda_i^{-1}$ for the smallest m (nonnull) eigenvalues λ_i of $\mathbf{L}$ and $r(\lambda_i) = 0$ for the remaining eigenvalues. It trivially penalizes the largest eigenvalues of $\mathbf{L}$, by "cutting" them off in the regularized Laplacian kernel.

Finally, notice that this graph PCA provides a nice interpretation to the "**Fiedler vector**" [24], [46], widely used for graph partitioning [13], [54]: The Fiedler vector simply contains the projections of the node vectors on the first principal component $\mathbf{u}_1$ of the graph. Indeed, the Fiedler vector corresponds to the smallest nontrivial eigenvector of the Laplacian matrix, which is also the first (with largest eigenvalue) eigenvector $\mathbf{u}_1$ of $\mathbf{L}^+$.

6 Experiments

6.1 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 a link to each movie watched by the corresponding person. Notice that, in this special case, the graph is bipartite. 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 0s and 1s). Moreover, our experiments do not take the `movie_category` set into account so that comparisons between the various scoring algorithms remain fair. Indeed, three standard scoring algorithms (i.e., maximum frequency, cosine, and nearest-neighbor algorithms) cannot naturally use the `movie_category` set to rank the movies.

6.1.1 Data Set

Our experiments were performed on a real movie database from the Web-based recommender system MovieLens (http://www.movielens.umn.edu). We used a sample of their database as suggested in [59]: Enough people (i.e., 943 people) were randomly selected to obtain 100,000 ratings (considering only persons that had rated 20 or more movies on a total of 1,682 movies).

A **preliminary experiment** was performed to tune the parameters of the scoring algorithms (we do not show the corresponding results in this paper), namely, parameter α of the Katz method, the number of dimensions for PCA CT, the similarity measure for the k nearest-neighbor algorithm, and the number k of neighbors for each scoring algorithm when using the indirect method (see Section 6.1.3). For this preliminary experiment, the database was divided into a `training set` and a `test set`. The `test set` contains

exactly 10 ratings for each of the 943 people (about 10,000 ratings), while the `training set` contains the remaining ratings (about 90,000 ratings). Thus, each rating belonging to the `test set` corresponds to a link that has been removed from the graph build from the complete database. The resulting pruned graph (without the links belonging to the `test set`) is the training graph.

In the **main experiment**, the data set was divided into n subsets (with $n = 10$) and the scoring algorithm was executed n times (n runs; n-fold cross-validation). In each run, 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 was computed on the n runs. 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.

For each person and each run, each scoring algorithm described in Section 6.1.4 computes, based on the `training set`, a ranked list of preferences about movies, expressed as similarities (scores) between people nodes and movie nodes. From that information, we retain a ranked list of all the movies *that the person has not watched*, according to the `training set` (predictions).

6.1.2 Performance Evaluation

To evaluate the scoring algorithms described in Section 6.1.4, we compared their performance using three measures: 1) the degree of agreement (which is a variant of Somers'D [63]), 2) a percentile score, and 3) a recall score. The `test set` contains, for each person and each run, a set of movies that the person has actually watched and that are not linked to that person in the `training set`. Those movies are part of the ranked list supplied by each scoring algorithm from the `training set`.

Degree of agreement. To compute the degree of agreement, we consider each pair of movies where one movie (say, $M1$) is in the `test set` for that person (the movie has actually been watched by that person) and the other movie (say, $M2$) is not in the `test set` nor in the `training set` for that person (the movie has not been watched by that person). A scoring algorithm ranks the pair in the correct order if, in the ranked list computed from the `training set`, movie $M1$ precedes movie $M2$. The individual degree of agreement is thus the percentage of pairs ranked in the correct order with respect to the total number of pairs [63]. The idea here is that the scoring algorithm should favor the movies that have indeed been watched by ranking them before those that have not been watched.

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

Percentile. The individual percentile score is simply the position (in percentages) that the median movie in the list from the `test set` occupies in the whole list of ranked movies computed from the `training set`. 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 percent. A random ranking would provide a percentile score of about 50 percent. This measure should be as low as possible for good performance (near 0 percent for a perfect model). The global percentile score is obtained by averaging the individual percentile scores of all persons and all runs.

Recall. The recall score is the average (on all persons) of the proportion (in percentages) of movies from the `test set` that appear among the *top* n of the ranked list from the `training set`, for some given n. This measure should be as high as possible for good performance. A recall score of 100 percent indicates that the scoring algorithm always positioned the movies in the `test set` among the *top* n of the ranked list. We computed the recall for the *top* 10 and the *top* 20 movies (for a total of 1,682 movies).

6.1.3 Direct Method versus Indirect Method for Recommendation

We have used three methods to determine which movies to suggest to a particular person, based on similarities between pairs of nodes:

Direct method. Use each scoring algorithm to compute the similarities between a given *person* and all the *movies*. Movies are simply presented in decreasing order of the similarity values.

User-based indirect method. First, use each scoring algorithm as in the direct method to compute the similarities (denoted $\mathrm{sim}(p_0, p)$) between a given *person* p_0 and all the other *persons* p; then, for p_0, compute from its k nearest neighbors (in the present case, nearest neighbors are persons) the *predicted value* (or preference) of each movie. The predicted value of movie m_0 for person p_0 is computed as a sum, weighted by $\mathrm{sim}(p_0, p)$, of the values of the link weight (0 or 1 here) of movie m_0 with the k persons p closest (according to $\mathrm{sim}(p_0, p)$) to person p_0:

$$\mathrm{pred}(p_0, m_0) = \frac{\sum_{p=1}^{k} \mathrm{sim}(p_0, p)\, a_{pm_0}}{\sum_{p=1}^{k} \mathrm{sim}(p_0, p)}, \tag{8}$$

where a_{pm_0} is 1 if person p watched movie m_0 and 0 otherwise. The sum is taken on the k nearest neighbors of p_0. The higher the predicted value $\mathrm{pred}(p_0, m_0)$, the stronger the recommendation that person p_0 should watch movie m_0. For each scoring algorithm, we systematically varied the number k of neighbors $(1, 2, \ldots, 10, 20, \ldots, 100)$ and we kept the value of k providing the best result. Notice that the value of k depends on the measure used to evaluate the performance (see Section 6.1.2).

Movie-based indirect method. This corresponds to the methodology proposed by Karypis in its SUGGEST approach [39]. First, use each scoring algorithm as in the direct method to compute the similarities (denoted $\mathrm{sim}(m_0, m)$) between a given *movie* m_0 and all the other *movies* m (in the present case, nearest neighbors are movies); then, for a given person p_0, compute from the k nearest neighbors of the movies watched by p_0 the predicted value of each movie. The predicted value of movie m_0 for person p_0 is computed as a sum, weighted by $\mathrm{sim}(m_0, m)$, of the values of the link weight (0 or 1) of person p_0 with the k nearest movies m of movie m_0:

$$\mathrm{pred}(p_0, m_0) = \frac{\sum_{m=1}^{k} \mathrm{sim}(m_0, m)\, a_{p_0 m}}{\sum_{m=1}^{k} \mathrm{sim}(m_0, m)}, \qquad (9)$$

where $a_{p_0 m}$ is defined as before. The movies are proposed to person p_0 in decreasing order of predicted values. As for the user-based indirect method, we optimized the number of neighbors. This way to suggest movies is equivalent to the SUGGEST method proposed in [39]. Notice that, if the algorithm provides a dissimilarity measure $\mathrm{dis}(i,j)$, we use $(\max - \mathrm{dis}(i,j))/(\max - \min)$ to convert it into a similarity measure.

6.1.4 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: simple k-nearest neighbors techniques, cosine coefficient, Katz' method, and the shortest-path algorithm. We also include the matrix-forest-based similarity measure proposed in [15]. We now describe these algorithms in more detail.

Maximum-frequency (MaxF). This scoring algorithm simply ranks the movies by the number of persons who watched them. In other words, 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 MaxF 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 component analysis based on ECTD (PCA CT). In Section 5, we showed that, based on the eigenvector decomposition of $\mathbf{L}^+$, the nodes can be mapped into a new Euclidean space (with more than 2,600 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 2,620 by a 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 needed 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 $(\mathrm{sim}(i,j) = l^+_{ij})$ 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 $\mathbf{L}^+$ (cos^+). This scoring algorithm computes similarities from (6) to rank the movies.

k-nearest neighbors (kNN). This scoring algorithm can be represented by the following rule: To classify a new item, choose the most frequent class of the k-nearest examples in the `training set` as measured by a similarity coefficient. Using a nearest-neighbor technique requires a measure of "closeness" or "similarity." The choice of a similarity measure (see [38]) includes to consider the nature of the variables (discrete, continuous, and binary), the scales of measurement (nominal, ordinal, interval, and ratio), and specific knowledge about the subject matter.

We now detail the procedure used for performing a k-nearest neighbors in the *user-based indirect method* (see Section 6.1.3). The procedure used for performing a k-nearest neighbors 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, i.e., watching a particular movie. 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 neighbors of person i are computed by taking the k nearest $\mathbf{v}_j$ according to a given similarity measure s between binary vectors, $\mathrm{sim}(i,j) = s(\mathbf{v}_i, \mathbf{v}_j)$. We performed systematic comparisons (preliminary experiment) between eight different such measures s (listed in [38, p. 674]) and for different values of $k\ (= 1, 2, \ldots, 10, 20, \ldots, 100)$. The best scores (for all the performance measures) were obtained with the "a ratio of 1-1 matches to mismatches." Notice that the k-nearest neighbors is an indirect method that cannot be used in the direct way.

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 $\mathrm{sim}(i,j) = \mathrm{cosine}(i,j) = (\mathbf{v}_i{}^{\mathrm{T}}\mathbf{v}_j)/(\|\mathbf{v}_i\|\,\|\mathbf{v}_j\|)$. We again systematically varied the number k of neighbors $(= 1, 2, \ldots, 10, 20, \ldots, 100)$. The cosine coefficient algorithm is also an indirect method that cannot be used in the direct method.

Katz (Katz). This similarity index has been proposed in the social sciences field and has been recently rediscovered in the context of collaborative recommendation [35] and kernel methods, where it is known as the von Neumann kernel [60]. Katz proposed in [40] 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

$$\mathbf{K} = \alpha\mathbf{A} + \alpha^2\mathbf{A}^2 + \ldots + \alpha^n\mathbf{A}^n + \ldots = (\mathbf{I} - \alpha\mathbf{A})^{-1} - \mathbf{I}, \tag{10}$$

where $\mathbf{A}$ is the adjacency matrix and α is a positive constant which has the force of a probability of effectiveness of a single link ($\mathrm{sim}(i,j) = k_{ij} = [\mathbf{K}]_{ij}$). A n-step chain or path, then, has a probability α^n 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 (preliminary experiment) the value of α ($\alpha = (0.05, 0.10, \ldots, 0.95) * (\mathrm{spectral\ radius})^{-1}$) and we present only the results obtained by the best model (namely, $\alpha = 0.05 * (\mathrm{spectral\ radius})^{-1}$).

Matrix-forest-based algorithm (MFA). The similarity matrix introduced by Chebotarev and Shamis in [15], [17] is

$$\mathbf{T} = (\mathbf{I} + \mathbf{L})^{-1}, \tag{11}$$

where $\mathbf{L}$ is the Laplacian matrix. This similarity measure has an interesting interpretation in terms of the matrix-forest theorem [15], [17]. 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 [15], [17] 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 ($\mathrm{sim}(i,j) = t_{ij} = [\mathbf{T}]_{ij}$), having the natural properties of a similarity (triangular property for similarities, among others [16]). It has recently been rediscovered and used in the context of documents ranking [37]. 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 $\lambda_i^{\mathrm{MFA}} = \lambda_i^{\mathbf{L}^+}/(1 + \lambda_i^{\mathbf{L}^+})$.

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 percent because of the large number of ties, and the detailed results are of little interest.

6.2 Cross-Validation Results

Thus, for each run of the cross-validation and for each person, we first select the movies that have not been watched, according to the `training set`. Then, we rank them according to all the described scoring algorithms and the methods to use them (direct, user-based indirect, or movie-based indirect method). 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.

6.2.1 Results and Discussion

All the results are summarized in Table 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 runs is also reported, as well as the optimal number of neighbors (Neighbors), when applicable.

Table 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 MFA. Notice that we use a paired t-test to determine if there is a significant difference (with a p-value smaller than 0.01) 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, based on the t-test.

In the **user-based indirect method**, the best results are obtained by $\mathbf{L}^+$ and MFA. 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 MFA 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 1 shows that $\mathbf{L}^+$, kNN, Cosine, $\cos^+$, and MFA 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 way (as proposed in the SUGGEST method, [39]), when considering the degree of agreement or the percentile, by $\mathbf{L}^+$ and MFA, used in the user-based indirect way, when considering the recall scores (considering either the top 10 or the top 20). We also observe that the user-based indirect method provides better recommendations than the movie-based indirect method for $\mathbf{L}^+$ and MFA, and for both kNN and Cosine, but only for recall scores.

The dissimilarity measures (i.e., CT, One-way, and Return) are clearly less efficient (with the exception of the Return in the user-based indirect method) and they seem to lack stability (the results are very sensitive to the method used). We observe that CT and One-way give better results in the direct method than in the indirect one and that their direct recommendations are very similar to simply recommending the most popular movies (as determined by MaxF). That both measures would be dominated by the stationary distribution was also suggested in [10], where it is shown that the commute time is highly sensitive to the degree of the nodes (which is equal, up to a scaling factor, to the stationary distribution of a simple random walk on the graph).

The fact that the inner products ($\mathbf{L}^+$, $\cos^+$, and MFA) 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

TABLE 1
Average Results Obtained by Performing a 10-Fold Cross-Validation, for the Various Scoring Algorithms and for the Three Methods Defined to Use Them (Direct, User-Based Indirect, and Movie-Based Indirect)

Direct method (in %)											
	MaxF	CT	PCA CT	One-way	Return	$\mathbf{L}^+$	$\cos^+$	kNN	Cosine	Katz	MFA
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

we observe in information retrieval. This result was also pointed out by Brand [10].

In conclusion, three similarity measures provide very good and stable performance: $\mathbf{L}^+$, MFA, and the simple nearest-neighbor technique (kNN). We also clearly observe that the most efficient scoring algorithms ($\mathbf{L}^+$, MFA) perform better with the indirect method than with the direct one. Which of the two indirect methods (user-based or movie-based) performs best is not clear, though. Indeed, kNN provides slightly better results in the movie-based indirect method, while the opposite holds for $\mathbf{L}^+$ and MFA.

6.2.2 Correlations between the Ranking Algorithms

The Kendall rank-order correlation coefficient provides a measure of the degree of association between two sets of rankings (see [63] for details; its range is [0,1] with 1 corresponding to a perfect association and 0.5 corresponding to no association at all). Table 2 shows the average correlations between the rankings provided by all the ranking algorithms.

First, 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 further observe that, as discussed 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 MFA are negatively correlated with MaxF. Thus, these methods do not always favor bestsellers.

6.2.3 Computing Times

We compared computing times (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). We did not try to exploit for any algorithm the sparseness of the adjacency matrix $\mathbf{A}$.

Table 3 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 the scoring algorithms were implemented in Matlab. We used, in order to compute the similarity matrices, (4) for the average commute time, the method suggested by Kemeny and Snell ([41, p. 218]) for the average first-passage times (one-way and return), (3) for $\mathbf{L}^+$ and the derived $\cos^+$, and (10) and (11) (both by inverting the matrix in Matlab) for, respectively, Katz and MFA similarity matrices. Notice also that, for the PCA, 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.

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 MFA.

7 Conclusions and Further Work

We have proposed a general procedure for computing similarities between elements of a database. It is based on a Markov-chain model of random walk through a graph representation of the database. More precisely, we compute

TABLE 2
Correlation Matrix Containing the Kendall Score for Each Pair of Scoring Algorithms and for the Three Methods Defined to Use Them (Direct, User-Based Indirect, and Movie-Based Indirect)

	CT	PCA CT	One-way	Return	$\mathbf{L}^+$	$\cos^+$	kNN	Cosine	Katz	MFA
Direct method										
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

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 of a connected graph. These similarity measures can be used in order to compare items belonging to database tables that are not necessarily directly connected. They rely on the degree of connectivity between these elements. 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 graphs. However, the nice interpretation in terms of the pseudoinverse of the Laplacian matrix does not apply in this case (still, see [1], [2] for potential generalizations to directed graphs).

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 in [42], the proposed 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 performances: the pseudoinverse of the Laplacian matrix, the matrix-forest-based similarity measure, and the simple nearest-neighbor technique (the latter when used in an item-based indirect method). However, for the nearest neighbor, two parameters need to be adjusted: the number of neighbors and the similarity between binary vectors, while the Laplacian pseudoinverse and the matrix-forest-based measures do not need any parameter tuning in the direct method and only need the number of neighbors in the indirect methods. Moreover, the pseudoinverse of the Laplacian matrix and the matrix-forest-based similarity measure are much more generic than the nearest neighbor 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 there are elements in the database. For large databases, we have to rely on iterative formulas, 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 various kernels on a graph that have been proposed in the literature on the same collaborative recommendation task [26]. Finally, we are working on generalizations to directed weighted graphs for which the weights are not necessarily positive.

TABLE 3
Time (in Seconds) Needed to Compute Predictions for All the Nonwatched Movies and All the Users

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

APPENDIX A
PROOF OF THE MAIN RESULTS

A.1 SOME USEFUL PROPERTIES OF THE PSEUDOINVERSE OF THE LAPLACIAN MATRIX

A.1.1 $\mathbf{L}^+$ Is Symmetric

Since $\mathbf{L}$ is symmetric and, for any matrix $\mathbf{M}$, $(\mathbf{M}^{\mathrm{T}})^+ = (\mathbf{M}^+)^{\mathrm{T}}$ (see [5]), we easily obtain $\mathbf{L}^+ = (\mathbf{L}^{\mathrm{T}})^+ = (\mathbf{L}^+)^{\mathrm{T}}$. Therefore, $\mathbf{L}^+$ is symmetric.

A.1.2 $\mathbf{L}^+$ Has Rank $n-1$ and Is Doubly Centered

An EP matrix $\mathbf{M}$ is a matrix that 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 [5, p. 253]). In particular, the following property of EP matrices is worth mentioning: 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 $\mathbf{L}^+$.

In particular, this implies that

1. $\mathbf{L}^+$ has rank $n-1$ and has the same null space as $\mathbf{L}$: $\mathbf{L}^+\mathbf{e} = 0$ $(\mathbf{e} = [\mathbf{1}, \mathbf{1}, \ldots, \mathbf{1}]^{\mathrm{T}}$ is the eigenvector associated to $\lambda_n = 0)$.
2. The previous property shows 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 [55, chapter 10], for a discussion of this topic).

Other properties of EP-matrices are described in [5] or [12].

A.1.3 $\mathbf{L}^+$ 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.

APPENDIX B

COMPUTATION OF THE AVERAGE FIRST-PASSAGE TIME/COST IN TERMS OF $\mathbf{L}^+$

Let us start from (2), rewritten here as

$$o(k|i) \;=\; r_i + \sum_{j=1}^{n} p_{ij} o(k|j), \text{ for } i \neq k, \tag{12}$$

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 $\mathbf{r}$. The objective is to find a closed-form solution for $o(k|i)$.

Let us renumber the states so that state k becomes state n, the last state of the Markov model, and rewrite (12) in matrix form. The equation will refer to vectors and matrices where the nth row and the nth column have been deleted; 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. We thus obtain $\widehat{\mathbf{o}} = \widehat{\mathbf{r}} + \widehat{\mathbf{P}}\widehat{\mathbf{o}} = \widehat{\mathbf{r}} + \widehat{\mathbf{D}}^{-1}\widehat{\mathbf{A}}\widehat{\mathbf{o}}$, 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 premultiply 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

$$\widehat{\mathbf{o}} = \widehat{\mathbf{L}}^{-1}\widehat{\mathbf{b}} = \widehat{\mathbf{L}}^{+}\widehat{\mathbf{b}} \tag{13}$$

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 be done thanks to the 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., [5], [8]).

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, e.g., [5], [8]) allowing us to compute the pseudoinverse of a matrix $\mathbf{M}_n = [\mathbf{M}_{n-1}\ \mathbf{a}]$ in terms of the pseudoinverse of $\mathbf{M}_{n-1}$:

$$\mathbf{M}_n^+ = [\,\mathbf{M}_{n-1} \quad \mathbf{a}\,]^+ = \begin{bmatrix} \mathbf{M}_{n-1}^+ - \mathbf{d}\mathbf{b}^{\mathrm{T}} \\ \mathbf{b}^{\mathrm{T}} \end{bmatrix}, \tag{14}$$

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

$$\mathbf{b}^{\mathrm{T}} = \begin{cases} \mathbf{c}^+ & \text{if } \mathbf{c} \neq \mathbf{0} \\ (1 + \mathbf{d}^{\mathrm{T}}\mathbf{d})^{-1}\mathbf{d}^{\mathrm{T}}\mathbf{M}_{n-1}^+ & \text{if } \mathbf{c} = \mathbf{0}, \end{cases} \tag{15}$$

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 (14)

$$\mathbf{M}_n^+ = \begin{bmatrix} \widehat{\mathbf{L}}^+ + \widehat{\mathbf{e}}\mathbf{b}^{\mathrm{T}} \\ \mathbf{b}^{\mathrm{T}} \end{bmatrix}. \tag{16}$$

By looking carefully at (16), 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 [29])). 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 $\mathbf{L}^{+}$:

$$\begin{aligned}\widehat{l}_{ij}^{+} &= [\mathbf{M}_n^{+}]_{ij} - [\mathbf{M}_n^{+}]_{nj} \\ &= [\mathbf{L}^{+}]_{ij} - [\mathbf{L}^{+}]_{in} - [\mathbf{L}^{+}]_{nj} + [\mathbf{L}^{+}]_{nn} \\ &= l_{ij}^{+} - l_{in}^{+} - l_{nj}^{+} + l_{nn}^{+}.\end{aligned} \tag{17}$$

By substituting (17) in (13), we obtain

$$\begin{aligned}o(n|i) &= \sum_{j=1}^{n-1} \widehat{l}_{ij}^{-1} b_j = \sum_{j=1}^{n-1} \Big(l_{ij}^{+} - l_{in}^{+} - l_{nj}^{+} + l_{nn}^{+} \Big) b_j \\ &= \sum_{j=1}^{n} \Big(l_{ij}^{+} - l_{in}^{+} - l_{nj}^{+} + l_{nn}^{+} \Big) b_j\end{aligned}$$

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

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

$$o(k|i) = \sum_{j=1}^{n} \Big(l_{ij}^{+} - l_{ik}^{+} - l_{kj}^{+} + l_{kk}^{+} \Big) b_j, \tag{18}$$

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,

$$m(k|i) = \sum_{j=1}^{n} \Big(l_{ij}^{+} - l_{ik}^{+} - l_{kj}^{+} + l_{kk}^{+} \Big) d_{jj}. \tag{19}$$

Appendix C

Computation of the Average Commute Time in Terms of $\mathbf{L}^{+}$

Since we already have the formula for the average first-passage time (19), computing the average commute time is trivial:

$$\begin{aligned}n(i,j) &= m(j|i) + m(i|j) \\ &= \sum_{k=1}^{n} \Big(l_{ik}^{+} - l_{ij}^{+} - l_{jk}^{+} + l_{jj}^{+} \Big) d_{kk} \\ &\quad + \sum_{k=1}^{n} \Big(l_{jk}^{+} - l_{ji}^{+} - l_{ik}^{+} + l_{ii}^{+} \Big) d_{kk} \\ &= \Big(l_{ii}^{+} + l_{jj}^{+} - 2 l_{ij}^{+} \Big) \sum_{k=1}^{n} d_{kk} \\ &= V_G \Big(l_{ii}^{+} + l_{jj}^{+} - 2 l_{ij}^{+} \Big).\end{aligned} \tag{20}$$

Equation (20) can be put in matrix form,

$$n(i,j) = V_G \, (\mathbf{e}_i - \mathbf{e}_j)^{\mathrm{T}} \mathbf{L}^{+} (\mathbf{e}_i - \mathbf{e}_j) \tag{21}$$

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

Appendix D

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, [49]). Hence, by using the transformation

$$\begin{cases} \mathbf{e}_i = \mathbf{U} \mathbf{x}_i \\ \mathbf{x}'_i = \mathbf{\Lambda}_i^{1/2} \mathbf{x}_i, \end{cases} \tag{22}$$

we obtain

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

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 (22) 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 $\sqrt{\lambda_1} u_1^1, \sqrt{\lambda_1} u_2^1, \ldots, \sqrt{\lambda_1} u_n^1$, and correspond to the principal eigenvector of $\mathbf{L}^{+}$ multiplied by $\sqrt{\lambda_1}$. The idea developed in Section 5 is thus to discard the axes (eigenvectors) corresponding to the smallest eigenvalues of $\mathbf{L}^{+}$.

Appendix E

$\mathbf{L}^{+}$ Is a Kernel

In this section, we show that $\mathbf{L}^{+}$ is a valid kernel or Gram matrix (see, for instance, [60], [61]). Indeed, $\mathbf{L}^{+}$ is the matrix containing the inner products of the transformed vectors $\mathbf{x}'_i$:

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

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

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

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:

$$\mathbf{X}' = \left[\mathbf{x}'_1, \mathbf{x}'_2, \ldots, \mathbf{x}'_n\right]^{\mathrm{T}}, \tag{23}$$

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

APPENDIX F
LINKS WITH PCA

We will now show that this decomposition is similar to PCA 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, $\mathbf{x}'^{\mathrm{T}}_i$, on each row (see (23)), we easily deduce from (22) 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 in fact the basis vectors of the transformed space. 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 [57].

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.

ACKNOWLEDGMENTS

The authors thank P. Van Dooren and V. Blondel from the "Département d'Ingénierie Mathématique" as well as P. Dupont from the "Département d'Ingénierie Informatique," at the Université catholique de Louvain, for insightful discussions. Part of this work has been funded by projects with the Belgian Federal Police, the "Région wallonne," and the "Région de Bruxelles-Capitale."

REFERENCES

[1] R. Agaev and P. Chebotarev, "The Matrix of Maximum Out Forests of a Digraph and Its Applications," *Automation and Remote Control,* vol. 61, no. 9, pp. 1424-1450, 2000.

[2] R. Agaev and P. Chebotarev, "Spanning Forests of a Digraph and Their Applications," *Automation and Remote Control,* vol. 62, no. 3, pp. 443-466, 2001.

[3] R. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval.* Addison-Wesley, 1999.

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

[5] S. Barnett, *Matrices: Methods and Applications.* Oxford Univ. Press, 1992.

[6] M. Belkin and P. Niyogi, "Laplacian Eigenmaps and Spectral Techniques for Embedding and Clustering," *Advances in Neural Information Processing Systems,* T.G. Dietterich, S. Becker, and Z. Ghahramani, eds., vol. 14, pp. 585-591, MIT Press, 2001.

[7] M. Belkin and P. Niyogi, "Laplacian Eigenmaps for Dimensionality Reduction and Data Representation," *Neural Computation,* vol. 15, pp. 1373-1396, 2003.

[8] A. Ben-Israel and T. Greville, *Generalized Inverses: Theory and Applications,* second ed. Springer-Verlag, 2003.

[9] I. Borg and P. Groenen, *Modern Multidimensional Scaling: Theory and Applications.* Springer, 1997.

[10] M. Brand, "A Random Walks Perspective on Maximizing Satisfaction and Profit," *Proc. 2005 SIAM Int'l Conf. Data Mining,* 2005.

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

[12] S. Campbell and C. Meyer, *Generalized Inverses of Linear Transformations.* Pitman Publishing Company, 1979.

[13] T. Chan, P. Ciarlet, and W. Szeto, "On the Optimality of the Median Cut Spectral Bisection Graph Partitioning Method," *SIAM J. Scientific Computing,* vol. 18, no. 3, pp. 943-948, 1997.

[14] 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," *Proc. Ann. ACM Symp. Theory of Computing,* pp. 574-586, 1989.

[15] P. Chebotarev and E. Shamis, "The Matrix-Forest Theorem and Measuring Relations in Small Social Groups," *Automation and Remote Control,* vol. 58, no. 9, pp. 1505-1514, 1997.

[16] P. Chebotarev and E. Shamis, "On a Duality between Metrics and S-Proximities," *Automation and Remote Control,* vol. 59, no. 4, pp. 608-612, 1998.

[17] P. Chebotarev and E. Shamis, "On Proximity Measures for Graph Vertices," *Automation and Remote Control,* vol. 59, no. 10, pp. 1443-1459, 1998.

[18] K. Cheung, K. Tsui, and J. Liu, "Extended Latent Class Models for Collaborative Recommendation," *IEEE Trans. Systems, Man, and Cybernatics. Part A: Systems and Humans,* vol. 34, pp. 143-148, 2004.

[19] F.R. Chung, *Spectral Graph Theory,* Am. Math. Soc., 1997.

[20] C. Ding, "Spectral Clustering," *Tutorial presented at the 16th European Conf. Machine Learning (ECML '05),* 2005.

[21] P.G. Doyle and J.L. Snell, *Random Walks and Electric Networks.* The Math. Assoc. of Am., 1984.

[22] S. Dzeroski, "Multi-Relational Data Mining: An Introduction," *ACM SIGKDD Explorations Newsletter,* vol. 5, no. 1, pp. 1-16, 2003.

[23] C. Faloutsos, K. McCurley, and A. Tomkins, "Fast Discovery of Connection Subgraphs," *Proc. 10th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining,* pp. 118-127, 2004.

[24] M. Fiedler, "A Property of Eigenvectors of Nonnegative Symmetric Matrices and its Applications to Graph Theory," *Czechoslovak Math. J.,* vol. 25, no. 100, pp. 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," *Proc. 2005 IEEE/WIC/ACM Int'l Joint Conf. Web Intelligence,* pp. 550-556, 2005.

[26] F. Fouss, L. Yen, A. Pirotte, and M. Saerens, "An Experimental Investigation of Graph Kernels on Two Collaborative Recommendation Tasks," submitted for publication.

[27] F. Gobel and A. Jagers, "Random Walks on Graphs," *Stochastic Processes and Their Applications,* vol. 2, pp. 311-336, 1974.

[28] J. Gower and P. Legendre, "Metric and Euclidean Properties of Dissimilarities Coefficients," *J. Classification,* vol. 3, pp. 5-48, 1986.

[29] F.A. Graybill, *Matrices with Applications in Statistics.* Wadsworth Int'l Group, 1983.

[30] A. Greenbaum, *Iterative Methods for Solving Linear Systems.* Soc. for Industrial and Applied Math., 1997.

[31] D. Harel and Y. Koren, "On Clustering Using Random Walks," *Proc. Conf. Foundations of Software Technology and Theoretical Computer Science,* pp. 18-41, 2001.

[32] J. Herlocker, J. Konstan, L. Terveen, and J. Riedl, "Evaluating Collaborative Filtering Recommender Systems," *ACM Trans. Information Systems,* vol. 22, no. 1, pp. 5-53, 2004.

[33] I. Herstein and D. Winter, *Matrix Theory and Linear Algebra.* Maxwell Macmillan International Editions, 1988.

[34] N.-D. Ho and P.V. Dooren, "On the Pseudo-Inverse of the Laplacian of a Bipartite Graph," *Applied Math. Letters,* vol. 18, no. 8, pp. 917-922, 2005.

[35] Z. Huang, H. Chen, and D. Zeng, "Applying Associative Retrieval Techniques to Alleviate the Sparsity Problem in Collaborative Filtering," *ACM Trans. Information Systems,* vol. 22, no. 1, pp. 116-142, 2004.

[36] D. Isaacson and R. Madsen, *Markov Chains Theory and Applications.* John Wiley and Sons, 1976.

[37] T. Ito, M. Shimbo, T. Kudo, and Y. Matsumoto, "Application of Kernels to Link Analysis: First Results," *Proc. Second Workshop Mining Graphs, Trees, and Sequences, ECML/PKDD,* pp. 13-24, 2004.
[38] R. Johnson and D. Wichern, *Applied Multivariate Statistical Analysis,* fifth ed. Prentice Hall, 2002.
[39] G. Karypis, "Evaluation of Item-Based Top-N Recommendation Algorithms," *Proc. 10th Int'l Conf. Information and Knowledge Management,* pp. 247-254, 2001.
[40] L. Katz, "A New Status Index Derived from Sociometric Analysis," *Psychmetrika,* vol. 18, no. 1, pp. 39-43, 1953.
[41] J.G. Kemeny and J.L. Snell, *Finite Markov Chains.* Springer-Verlag, 1976.
[42] D.J. Klein and M. Randic, "Resistance Distance," *J. Math. Chemistry,* vol. 12, pp. 81-95, 1993.
[43] J.M. Kleinberg, "Authoritative Sources in a Hyperlinked Environment," *J. ACM,* vol. 46, no. 5, pp. 604-632, 1999.
[44] R.I. Kondor and J. Lafferty, "Diffusion Kernels on Graphs and Other Discrete Structures," *Proc. 19th Int'l Conf. Machine Learning,* pp. 315-322, 2002.
[45] C.D. Meyer, "The Role of the Group Generalized Inverse in the Theory of Finite Markov Chains," *SIAM Rev.,* vol. 17, pp. 443-464, 1975.
[46] B. Mohar, "Laplace Eigenvalues of Graphs—A Survey," *Discrete Math.,* vol. 109, pp. 171-183, 1992.
[47] B. Nadler, S. Lafon, R. Coifman, and I. Kevrekidis, "Diffusion Maps, Spectral Clustering and Eigenfunctions of Fokker-Planck Operators," *Advances in Neural Information Processing Systems 18,* pp. 955-962, 2006.
[48] M. Newman, "A Measure of Betweenness Centrality Based on Random Walks," *Social Networks,* vol. 27, no. 1, pp. 39-54, 2005.
[49] B. Noble and J. Daniels, *Applied Linear Algebra,* third ed. Prentice-Hall, 1988.
[50] J. Norris, *Markov Chains.* Cambridge Univ. Press, 1997.
[51] L. Page, S. Brin, R. Motwani, and T. Winograd, "The Pagerank Citation Ranking: Bringing Order to the Web," technical report, Computer System Laboratory, Stanford Univ., 1998.
[52] C. Palmer and C. Faloutsos, "Electricity Based External Similarity of Categorical Attributes," *Proc. Seventh Pacific-Asia Conf. Knowledge Discovery and Data Mining (PAKDD '03),* pp. 486-500, 2003.
[53] E. Parzen, *Stochastic Processes.* Holden-Day, 1962.
[54] A. Pothen, H. Simon, and K.-P. Liou, "Partitioning Sparse Matrices with Eigenvectors of Graphs," *SIAM J. Matrix Analysis and Applications,* vol. 11, no. 3, pp. 430-452, 1990.
[55] C. Rao and S. Mitra, *Generalized Inverse of Matrices and Its Applications.* John Wiley and Sons, 1971.
[56] Y. Saad, *Iterative Methods for Sparse Linear Systems.* Soc. for Industrial and Applied Math., 2000.
[57] M. Saerens and F. Fouss, A Novel Way of Computing Similarities between Nodes of a Graph, with Application to Collaborative Filtering and Subspace Projection of the Graph Nodes, working paper, IAG, Université catholique de Louvain, 2006.
[58] M. Saerens, F. Fouss, L. Yen, and P. Dupont, "The Principal Components Analysis of a Graph, and Its Relationships to Spectral Clustering," *Proc. 15th European Conf. Machine Learning (ECML '04),* pp. 371-383, 2004.
[59] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Recommender Systems for Large-Scale E-Commerce: Scalable Neighborhood Formation Using Clustering," *Proc. Fifth Int'l Conf. Computer and Information Technology,* 2002.
[60] B. Scholkopf and A. Smola, *Learning with Kernels.* The MIT Press, 2002.
[61] J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis.* Cambridge Univ. Press, 2004.
[62] J. Shi and J. Malik, "Normalised Cuts and Image Segmentation," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 22, no. 8, pp. 888-905, Aug. 2000.
[63] S. Siegel and J. Castellan, *Nonparametric Statistics for the Behavioral Sciences,* second ed. McGraw-Hill, 1988.
[64] A.J. Smola and R. Kondor, "Kernels and Regularization on Graphs," *Proc. Conf. Learning Theory (COLT) and Kernels Workshop,* M. Warmuth and B. Schölkopf, eds., 2003.
[65] S. Wasserman and K. Faust, *Social Network Analysis: Methods and Applications.* Cambridge Univ. Press, 1994.
[66] S. White and P. Smyth, "Algorithms for Estimating Relative Importance in Networks," *Proc. Ninth ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining,* pp. 266-275, 2003.
[67] D. Zhou, J. Huang, and B. Scholkopf, "Learning from Labeled and Unlabeled Data on a Directed Graph," *Proc. 22nd Int'l Conf. Machine Learning,* pp. 1041-1048, 2005.

François Fouss received the BS degree in management sciences in 2001 and the MS degree in information systems in 2002, both from the Université catholique de Louvain (UCL), Belgium. He is now preparing a PhD thesis in the field of machine learning and data mining. His main research areas are collaborative recommendations, graph mining, and classification.

Alain Pirotte is a professor of computing science and information management at the Université catholique de Louvain (UCL) and at the Université Libre de Bruxelles (ULB), Belgium. Earlier, he was a researcher in industry for more than 20 years. His interests include database management, information modeling, the methodology for software development, and cooperation with developing countries. He is a member of the IEEE.

Jean-Michel Renders received the PhD Degree from the Université Libre de Bruxelles (ULB), Belgium, in 1993. His main research interests include machine learning techniques applied to statistical natural language processing and text mining. He is currently a senior scientist at Xerox Research Centre Europe, where he is actively involved in document categorization, multimedia and multilingual information retrieval, and entity extraction projects. He is a member of the PASCAL European network of excellence http://www.pascal-network.org/Network/Sites/54/ (Pattern Analysis, Statistical modelling and ComputAtional Learning).

Marco Saerens received the PhD degree in engineering from the Université Libre de Bruxelles (ULB), Belgium, in 1991, and after graduation joined the IRIDIA Laboratory (the artificial intelligence laboratory, ULB) as a research assistant. While remaining a part-time researcher at IRIDIA, he then worked as a senior researcher in the R and D department of various industries, mainly in the fields of speech recognition, data mining, and artificial intelligence. In 2002, he joined the Universite catholique de Louvain (UCL) as a professor in computing science. His main research interests include artificial intelligence, machine learning, data mining, pattern recognition, and speech/language processing. He is a member of the IEEE.