Exact and Approximate Graph Matching Using Random Walks

Marco Gori, *Fellow*, *IEEE*, Marco Maggini, *Member*, *IEEE Computer Society*, and Lorenzo Sarti

Abstract—In this paper, we propose a general framework for graph matching which is suitable for different problems of pattern recognition. The pattern representation we assume is at the same time highly structured, like for classic syntactic and structural approaches, and of subsymbolic nature with real-valued features, like for connectionist and statistic approaches. We show that random walk based models, inspired by Google's PageRank, give rise to a spectral theory that nicely enhances the graph topological features at node level. As a straightforward consequence, we derive a polynomial algorithm for the classic graph isomorphism problem, under the restriction of dealing with *Markovian spectrally distinguishable graphs* (MSD), a class of graphs that does not seem to be easily reducible to others proposed in the literature. The experimental results that we found on different test-beds of the TC-15 graph database show that the defined MSD class "almost always" covers the database, and that the proposed algorithm is significantly more efficient than top scoring VF algorithm on the same data. Most interestingly, the proposed approach is very well-suited for dealing with partial and approximate graph matching problems, derived for instance from image retrieval tasks. We consider the objects of the COIL-100 visual collection and provide a graph-based representation, whose node's labels contain appropriate visual features. We show that the adoption of classic bipartite graph matching algorithms offers a straightforward generalization of the algorithm given for graph isomorphism and, finally, we report very promising experimental results on the COIL-100 visual collection.

Index Terms—Exact graph matching, approximate graph matching, random walks, PageRank, image retrieval.

1 Introduction

In the last three decades, the emphasis on the research in the area of pattern recognition has hovered pendulum-like from decision-theoretic to structured approaches. Decision-theoretic methods are essentially based on numerical features, which provide a flat representation of the pattern by means of an appropriate preprocessing. On the opposite, syntactic and structural pattern recognition and, additionally, artificial intelligence-based methods have been developed which emphasize the symbolic nature of patterns. Such different approaches to pattern recognition have given rise to the longstanding debate, which takes place also in the artificial intelligence community, between symbolic and subsymbolic models. However, both purely decision-theoretic and syntactical/structural approaches are limited when applied to many interesting real-world problems for opposite reasons and, therefore, different integration proposals of these approaches have recently emerged.

In this paper, we focus on graph matching by relying on a pattern representation which is at the same time highly structured, like for classic syntactic and structural approaches, and of subsymbolic nature with real-valued features, like for connectionist and statistic approaches. In a sense, the simplest form of graph matching yields for the case of nodes with no labels where one wants to construct the isomorphism between two given graphs. This problem is not only relevant for pattern recognition, but is also itself of great interest in theoretical computer science since, as yet, neither a polynomial algorithm has been exhibited nor proof has been given that the problem belongs to the class of NP-complete. Many efforts have been spent for devising graph isomorphism algorithms that are feasible under some practical conditions. A common approach is that of looking for an optimal representation of the search space and choosing effective pruning policies. For instance, Ullmann's algorithm [1] can be used for both graph and subgraph isomorphism detection, and exploits a backtracking method to prune efficiently the search space. Related methods were proposed in [2], [3] and a very efficient algorithm, known as VF [4], has been recently introduced which is based on a depth-first search strategy, with a set of rules to prune the search tree. Unfortunately, the solution of the graph isomorphism is not very useful in pattern recognition, where one is more interested in approximate matching and where the labels in the node may play an important role. In early inexact graph-matching works, the concept of string edit distance was extended to graphs [5]. Other recent techniques, based on nondeterministic paradigms were proposed. These methods are based on probabilistic relaxation [6], or on genetic algorithms [7]. Unfortunately, all these methods are generally intractable as the graph dimension increases, thus making it very hard the application to important real-world problems. Other methods try to characterize the global structural properties of graphs using the eigenvalues and eigenvectors of the adjacency matrix. The method proposed by Umeyama [8] exploits an eigen-decomposition to match graphs of the same size. Recently, Luo and Hancock [9] have extend the method

• *The authors are with DII—Università degli Studi di Siena, Via Roma, 56-53100 Siena, Italy. E-mail: {marco, maggini, sarti}@dii.unisi.it.*

Manuscript received 5 Jan. 2004; revised 5 Aug. 2004; accepted 1 Oct. 2004; published online 12 May 2005.
Recommended for acceptance by M. Basu.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org and reference IEEECS Log Number TPAMISI-0012-0104.

proposed by Umeyama in order to match graphs with structural errors and differences in graph-size. DePiero et al. [10] propose a node-to-node mapping which is based on the powers of the graph adjacency matrix $\boldsymbol{A}$. In so doing, the connectivity signature kept in $\boldsymbol{A}^m$ describes the number of paths from node i to node j of length m.

The matching method that we propose is a somehow related spectral theory based on Markov random walks, which is inspired by Google's PageRank. The notion of spectrum introduced in the theory is based on the probability of visiting the graph nodes while changing a "damping parameter," that defines the extent to which the random walk depends primarily on the node label or primarily on the graph topology. A significant consequence of the proposed theory is that the given notion of spectrum gives rise to a polynomial algorithm, referred to as Random Walk Algorithm (RW Algorithm), for the classic graph isomorphism problem, under the restriction of dealing with *Markovian spectrally distinguishable graphs* (MSD), a class that does not seem to be easily reducible to others proposed in the literature. In order to understand the relevance of this theoretical result, we carried experiments on the IAPR TC-15 graph database.[1]

Our experiments on different test-beds of the TC-15 graph database show that the defined MSD class "almost always" covers the database, and that the proposed algorithm is significantly more efficient than top scoring VF algorithm on the same data. Interestingly, we show that the proposed approach is not only appropriate for graph isomorphism, but that it is also very well-suited for dealing with partial and approximate graph matching derived for instance from problems of image retrieval. The objects of the visual collection that we consider are properly represented by graphs, whose node's labels contain visual features. The retrieval is based on partial matching of graphs in a way that closely reminds us the proposed approach to graph isomorphism. Given any two graphs, they are regarded as the two components of a bipartite graph whose node's labels are enriched with the topological features derived from the mentioned spectral analysis. The adoption of classic bipartite graph matching algorithms, in which the match between any two nodes is based on a distance in this enriched label space, gives rise to very promising experimental results in the COIL-100 database.[2]

The paper is organized as follows: In the next section, the Markov random walk theory used to compute the node topological signatures is defined. In Section 3, the novel matching algorithms are described and, in Section 4, the experimental results are reported. Finally, in Section 5, some conclusions are drawn.

2 Random Walks and Graph Matching

The theory of Random Walks has been proposed as a framework to define models which compute the absolute relevance of pages (vertices) in a hyperlinked environment like the Web [11]. The relevance of a page depends on the topological properties of the links among the pages. A more general framework for this scheme, which takes into account also the page contents (labels attached to the vertices), was proposed in [12]. Thus, in this paper, we introduce a general framework to compute a set of topological signatures for each node in a graph, which can then be used to construct a mapping between the nodes of two graphs. This mapping allows us to perform graph and subgraph matching, either exactly or approximately.

A Random Walk (RW) on a given graph G is described by a probabilistic model which allows us to compute the probability $\boldsymbol{x}_p(t)$ of being located in each vertex p at time t. The probability distribution on all the vertices is represented by a vector $\boldsymbol{x}(t) = [x_1(t), \ldots, x_N(t)]'$, being N the number of vertices in the graph. The probabilities $x_p(t)$ are updated at each time step by considering a set of actions that can be taken. In particular, we assume that the random walker can move from node q to node p by following the arc (q,p) or by *jumping* directly to p. Thus, the probability distribution is updated using the following equation[3]

$$x_p(t+1) = \sum_{q \in \mathbf{G}} x(p|q,j) \cdot x(j|q) \cdot x_q(t) + \sum_{q \in pa(p)} x(p|q,l) \cdot x(l|q) \cdot x_q(t), \tag{1}$$

where $x(p|q,j)$ and $x(p|q,l)$ are the probabilities of moving from q to p by performing a jump or by traversing the arc (q,p), respectively, and $x(l|q)$ and $x(j|q)$ represent the bias between the two possible actions, i.e., *jump* and *follow an arc*. These parameters describe the RW behavior and can be chosen in order to extract different topological features of the graph. Moreover, since they represent probabilities, their values must be normalized such that

$$\forall q \in \mathbf{G}, \quad \sum_{p \in \mathbf{G}} x(p|q,j) = 1, \quad \sum_{p \in ch(q)} x(p|q,l) = 1, \quad x(j|q) = 1 - x(l|q). \tag{2}$$

The RW model can be described in matrix form as

$$\boldsymbol{x}(t+1) = (\boldsymbol{\Sigma} \cdot \boldsymbol{D}_{\mathrm{j}})' \boldsymbol{x}(t) + (\boldsymbol{\Delta} \cdot \boldsymbol{D}_{\mathrm{l}})' \boldsymbol{x}(t), \tag{3}$$

where $\boldsymbol{\Sigma}, \boldsymbol{\Delta} \in \mathbb{R}^{\mathrm{N,N}}$ collect the probabilities $x(p|q,j)$ and $x(p|q,l)$, respectively, and $\boldsymbol{D}_j, \boldsymbol{D}_l \in \mathbb{R}^{N,N}$ are diagonal matrices whose diagonal values are the probabilities $x(j|q)$ and $x(l|q)$. The entry (p,q) of matrix $\boldsymbol{\Delta}$ is not null only if the corresponding entry of the graph adjacency matrix A is equal to 1, i.e., if the vertices p and q are linked by an arc. By defining the transition matrix as $T = \left(\boldsymbol{\Sigma} \cdot \boldsymbol{D}_{\mathrm{j}} + \boldsymbol{\Delta} \cdot \boldsymbol{D}_{\mathrm{l}}\right)'$, (3) can be written as

$$\boldsymbol{x}(t+1) = \boldsymbol{T} \cdot \boldsymbol{x}(t). \tag{4}$$

1. The database can be downloaded at http://amalfi.dis.unina.it/graph/ along with related information.

2. The database can be accessed at http://www1.cs.columbia.edu/CAVE/research/softlib/coil-100.html.

3. Given a node p in the graph G, $pa(p)$ denotes the set of the parents of p and $ch(p)$ is the set of the children of p.

The signature of the graph nodes is obtained considering the steady state distribution $\boldsymbol{x}^\star$ of the Markov chain defined in (4). Notice that $\boldsymbol{T}$ is stable, since it is a stochastic matrix having its maximum eigenvalue equal to one and this guarantees that the process converges to $\boldsymbol{x}^\star$ (see, e.g., [13], chapter 4).

The RW model depends on the set of parameters collected in the matrices $\boldsymbol{\Sigma}$, $\boldsymbol{\Delta}$, $\boldsymbol{D}_j$, and $\boldsymbol{D}_l$. The entries of these matrices need to be defined in order to extract useful node signatures. In particular, when the graph is labeled, the parameters can be computed as a function of the labels of the nodes at each end of the arc. In the following sections, the two cases of unlabeled and labeled graphs are considered.

2.1 Unlabeled Graphs

In order to compute the node signatures, we define a RW for which the action bias probabilities $x(j|q)$ and $x(l|q)$ are independent of the node q. Thus, we consider a parameter $d \in (0,1)$ (*damping factor*) such that $x(l|p) = d$ and $x(j|p) = 1-d$. This parameter can be varied in order to obtain different topological signatures $\boldsymbol{x}^\star(d)$ for the graph. Moreover, the other model parameters are chosen by considering uniform probability distributions. The target for a jump is selected using a uniform probability distribution over all the N nodes of the graph, i.e., $x(p|j) = 1/N, \forall p \in \boldsymbol{G}$. Finally, we assume that all the arcs from node q have the same probability to be followed, i.e., $x(p|q,l) = 1/o_q$, where $o_q = |ch(q)|$ is the number of arcs outcoming from q. This requirement cannot be met by vertices without outcoming arcs (*sink* nodes). For these nodes, the RW behavior is set such that $x(j|p_{sink}) = 1$ and $x(l|p_{sink}) = 0$.

With these settings, (3) can be rewritten as

$$\boldsymbol{x}(t+1) = \frac{(1-d)}{N} \cdot \mathbb{1} + d \cdot \boldsymbol{W} \cdot \boldsymbol{x}(t), \tag{5}$$

where $\boldsymbol{W} = (\boldsymbol{\Theta}\boldsymbol{A})'$ being $\boldsymbol{A}$ the adjacency matrix of the graph and $\boldsymbol{\Theta}$ the diagonal matrix whose (p,p) element is the $1/o_p$, while $\mathbb{1}$ is a vector whose entries are all equal to one. Equation (5) is related to the PageRank [11] used by the Google search engine to compute the relevance of each page (vertex) in the Web (graph). The steady state distribution of the RW defined by this equation can be expressed as

$$\boldsymbol{x}^\star = \frac{1-d}{N}(I - d\boldsymbol{W})^{-1}\mathbb{1} = \frac{1-d}{N}\sum_{k=0}^{\infty} d^k W^k \cdot \mathbb{1}, \tag{6}$$

where we used the series expansion for $(I - dW)^{-1}$.

The signatures computed by the RW model have useful properties. First of all, the following proposition motivates the use of this signature for detecting the isomorphism between two graphs.

Proposition 1. *If two graphs $\boldsymbol{G}_1$ and $\boldsymbol{G}_2$ are isomorphic then $\boldsymbol{x}_1^\star = \boldsymbol{P}\boldsymbol{x}_2^\star$, where $\boldsymbol{P}$ is the permutation matrix such that $\boldsymbol{A}_1 = \boldsymbol{P}\boldsymbol{A}_2\boldsymbol{P}'$, being $\boldsymbol{A}_1, \boldsymbol{A}_2$ the adjacency matrices of $\boldsymbol{G}_1$ and $\boldsymbol{G}_2$, respectively.*

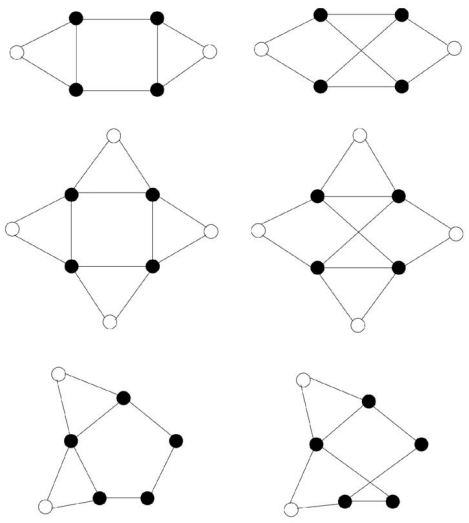

Fig. 1. Examples of graphs which have the same RW steady state. Black vertices denote a common isomorphic and regular subgraph.

Proof. Since $\boldsymbol{A}_1 = \boldsymbol{P}\boldsymbol{A}_2\boldsymbol{P}'$, it can be easily derived that $\boldsymbol{W}_1 = \boldsymbol{P}\boldsymbol{W}_2\boldsymbol{P}'$. If we consider the RW steady state $\boldsymbol{x}_1^\star$ given by (6), we obtain

$$\begin{aligned}\boldsymbol{x}_1^\star &= \frac{1-d}{N}(I - d\boldsymbol{W}_1)^{-1}\mathbb{1} \\ &= \frac{1-d}{N}(\boldsymbol{P}\boldsymbol{P}' - d\boldsymbol{P}\boldsymbol{W}_2\boldsymbol{P}')^{-1}\mathbb{1} \\ &= \frac{1-d}{N}\boldsymbol{P}(I - d\boldsymbol{W}_2)^{-1}\boldsymbol{P}'\mathbb{1} \\ &= \boldsymbol{P}\frac{1-d}{N}(I - d\boldsymbol{W}_2)^{-1}\mathbb{1} = \boldsymbol{P}\boldsymbol{x}_2^\star\end{aligned}$$

since $\boldsymbol{P}'\mathbb{1} = \mathbb{1}$. □

Unfortunately, the vice-versa is not always true. In fact, there exist not isomorphic graphs which, for any value of the parameter d, have the same RW steady state except for a permutation of the components (see, e.g., Fig. 1). Notice that undirected graphs can be processed considering that an undirected edge can be replaced by a pair of directed arcs with opposite directions.

Given a vector $D = [d_1, d_2, \ldots, d_N]$ collecting N distinct values for d such that $d_1 < d_2 < \ldots < d_N$, we can define the *graph discrete spectrum matrix* $\boldsymbol{Z}^\star(\boldsymbol{G}, D) \in \mathbb{R}^{N,N}$ as

$$\boldsymbol{Z}^\star(\boldsymbol{G}, D) = [\boldsymbol{x}^\star(d_1)| \ldots |\boldsymbol{x}^\star(d_N)].$$

When the matrix $\boldsymbol{Z}^\star(\boldsymbol{G}, D)$ is full rank, it provides a representation which encodes exactly the graph $\boldsymbol{G}$. In fact, the following proposition holds.

Proposition 2. *Let $\boldsymbol{Z}^\star(\boldsymbol{G}, D)$ be the graph discrete spectrum matrix of a graph $\boldsymbol{G}$ for a given monotonic sampling D and let $\mathbb{1}_N$ be a $N \times N$ matrix with all the entries equal to 1. If $rank(\boldsymbol{Z}^\star(\boldsymbol{G}, D)) = N$, then the graph $\boldsymbol{G}$ can be univocally reconstructed from $\boldsymbol{Z}^\star(\boldsymbol{G}, D)$ and the matrix $\boldsymbol{W}$ associated to $\boldsymbol{G}$ can be obtained by*

$$\boldsymbol{W} = [\boldsymbol{Z}^\star(\boldsymbol{G}, D) - \mathbb{1}_N(\boldsymbol{I} - \boldsymbol{D})] \cdot [\boldsymbol{Z}^\star(\boldsymbol{G}, D)\boldsymbol{D}]^{-1},$$

where D is a diagonal matrix whose (i,i) entry is set to d_i.

Proof. By rewriting (5) for the steady state vector and collecting the columns for the values of the parameter d in the vector D, we obtain

$$\boldsymbol{Z}^\star(\boldsymbol{G}, D) = \boldsymbol{W}\boldsymbol{Z}^\star(\boldsymbol{G}, D)\boldsymbol{D} + \mathbb{1}_N(I - \boldsymbol{D}).$$

Since $rank(\boldsymbol{Z}^\star(\boldsymbol{G}, D)) = rank(\boldsymbol{Z}^\star(\boldsymbol{G}, D)\boldsymbol{D}) = N$, the matrix $\boldsymbol{Z}^\star(G, D)D$ is invertible. Thus, we can solve the previous equation with respect to $\boldsymbol{W}$, which yields the thesis. □

The following theoretical result states that the rank of $\boldsymbol{Z}^\star(G, D)$ is related to the rank of the reachability matrix of the graph defined as $\boldsymbol{R} = [\mathbb{1}\ \boldsymbol{W}\mathbb{1}\ \boldsymbol{W}^2\mathbb{1}\ \ldots\ \boldsymbol{W}^{N-1}\mathbb{1}]$.

Proposition 3. *If $rank(\boldsymbol{R}) = r$, then $rank(\boldsymbol{Z}^\star(\boldsymbol{G}, D)) \leq r$ for any sampling D.*

Proof. This thesis derives straightforwardly from a well-known result in Linear System Theory concerning state reachability (see Theorem 5, [14, p. 272]) which is based on the Cayley-Hamilton theorem. In particular, from (6), the cited result states that $\boldsymbol{x}^\star(d) \in span(\boldsymbol{R})$ for any damping factor d and, thus, there exists a matrix $\boldsymbol{V}(D) \in \mathbb{R}^{N,N}$ such that

$$\boldsymbol{Z}^\star(\boldsymbol{G}, D) = \boldsymbol{R}\boldsymbol{V}(D)$$

from which the thesis derives straightforwardly. □

This result gives an upper bound to the rank of the matrix $\boldsymbol{Z}^\star(\boldsymbol{G}, D)$, which plays a crucial role in the idea we propose for graph isomorphism. Interestingly, Pasini [15] has recently proven that the relationship between $\boldsymbol{Z}^\star(\boldsymbol{G}, D)$ and the reachability matrix R is even closer. In the case in which $\boldsymbol{W}$ has distinct eigenvalues, Pasini has proven that $rank(\boldsymbol{R}) = rank(\boldsymbol{Z}^\star(\boldsymbol{G}, D))$. This has a strong experimental consequence, since it states that $rank(\boldsymbol{Z}^\star(\boldsymbol{G}, D))$ is independent of the choice of the sampling of the damping parameter d.

Finally, as stated in the following theorem, when $\boldsymbol{Z}^\star(\boldsymbol{G}, D)$ is full rank, it can be used to verify straightforwardly the isomorphism of any two graphs.

Theorem 1. *Given any two graphs $\boldsymbol{G}_1$ and $\boldsymbol{G}_2$, let $\boldsymbol{Z}^\star(\boldsymbol{G}_1, D)$ and $\boldsymbol{Z}^\star(\boldsymbol{G}_2, D)$ be their graph discrete spectrum matrices and assume that*

1. *there exists a permutation matrix P such that $\boldsymbol{Z}^\star(\boldsymbol{G}_2, D) = \boldsymbol{P} \cdot \boldsymbol{Z}^\star(\boldsymbol{G}_1, D)$ and*
2. *the matrix $\boldsymbol{Z}^\star(\boldsymbol{G}_1, D)$ is full rank,*

then $\boldsymbol{G}_1$ is isomorph to $\boldsymbol{G}_2$.

Proof. From the hypothesis that there exists P such that $\boldsymbol{Z}^\star(\boldsymbol{G}_2, D) = \boldsymbol{P} \cdot \boldsymbol{Z}^\star(\boldsymbol{G}_1, D)$ and, from Proposition 2, we derive

$$\begin{aligned} W_2 &= [\boldsymbol{P}\boldsymbol{Z}^\star(\boldsymbol{G}_1, D) - \boldsymbol{P}\mathbb{1}(I - \boldsymbol{D})P'] \cdot [\boldsymbol{P}\boldsymbol{Z}^\star(\boldsymbol{G}_1, D)\boldsymbol{D}]^{-1} \\ &= \boldsymbol{P}[\boldsymbol{Z}^\star(\boldsymbol{G}_1, D) - \mathbb{1}(I - \boldsymbol{D})] \cdot [\boldsymbol{Z}^\star(\boldsymbol{G}_1, D)\boldsymbol{D}]^{-1}P' \\ &= \boldsymbol{P} \cdot \boldsymbol{W}_1\boldsymbol{P}' \end{aligned}$$

that is $\boldsymbol{G}_1$ is isomorph to $\boldsymbol{G}_2$. □

The graphs for which the discrete spectrum matrix is full rank are referred to as *Markovian spectrally distinguishable graphs*. As already pointed out, this property is independent of the sampling of the damping parameter d and, in the case in which $\boldsymbol{W}$ has distinct eigenvalues, it can be established simply by inspecting the reachability matrix $\boldsymbol{R}$.

2.2 Labeled Graphs

Very frequently, in pattern recognition applications, graphs contain also information associated to each vertex (node labels). In this case the RW parameters can be computed as a function of the node labels. In the following, we propose two possible approaches to compute the probabilities $x(p|q, l)$ and $x(p|q, j)$. In the first case, the probability values are obtained using predefined functions, while in the second case they are estimated on a given data set. In both cases, we consider that the probabilities for the two actions are independent of the node by setting $x(l|p) = d$ and $x(j|p) = (1 - d),\ (0 < d < 1)$ as in the case of unlabeled graphs. This choice allows us to keep the damping factor d as a free parameter to extract different signatures on the same graph. In the following, we suppose that the label attached to the each node p of the graph is represented by a vector $L_p \in \mathbb{R}^m$.

Tendency functions. The *tendency* of following an arc from the vertex q to the vertex p is a function $f_l : \mathbb{R}^{2m} \rightarrow \mathbb{R}^+$ of the two node labels L_q and L_p. Using this function, we can define the probability of following a given arc from any node q as

$$x(p|q, l) = \frac{f_l(L_p, L_q)}{\sum_{i \in ch[q]} f_l(L_i, L_q)}.$$

Analogously, we can define the tendency of jumping from the vertex q to the vertex p as a function $f_j : \mathbb{R}^{2m} \rightarrow \mathbb{R}^+$ of the two node labels. Also in this case, the probability of jumping from q to p is obtained by normalizing the tendency function as

$$x(p|q, j) = \frac{f_j(L_p, L_q)}{\sum_{i \in \mathrm{G}} f_j(L_i, L_q)}.$$

Using the previous definitions, both $x(p|q, l)$ and $x(p|q, j)$ meet the normalization constraints required by the RW model except for the sink nodes. However, the solution described in Section 2.1 to deal with sinks can be applied also in this case.

The choice of the tendency functions plays a crucial role for the computation of the topological features. For instance, these functions can implement a sort of similarity function between the two labels. In this case, the RW will be more likely to follow arcs which link nodes storing similar labels. Further details about the suitable choice of tendency functions for graphs representing images will be described in Section 4.

Estimation. For a given task, a subset of graphs can be selected in order to provide a representative set of data on which the RW transition probabilities are estimated. Because of the hypothesis that the probabilities depend on the labels, we need to estimate the two distributions $prob((q, p)|L_q, L_p)$ and $prob(q|L_q)$. Then, we can use these distributions to define $x(p|q, l)$ and $x(p|q, j)$, respectively. In order to simplify the estimation of the parameters, we assume that the vertices are grouped into K clusters according to their labels. Thus, the RW behavior for any

vertex p is dependent only on the cluster which p belongs to. We can use a classical clustering techniques (e.g., K-means) on the labels associated to each vertex. By this hypothesis, the transition probabilities are estimated as

$$\begin{aligned} x(p|q,l) &= \frac{prob((q,p)|c_q,c_p)}{\sum_{k\in ch(q)} prob((q,k)|c_q,c_k)} \\ x(p|q,j) &= \frac{|c_p|}{\sum_{i=1}^{K}|c_i|}, \end{aligned} \tag{7}$$

where c_i is the cluster which node i belongs to, $|c_i|$ is the number of nodes in cluster c_i, and $prob((q,p)|c_q,c_p)$ is the probability of a link from a node belonging to cluster c_q to a node in cluster c_p.

3 Matching Algorithms

The vertex signatures computed using the RW models described in the previous section can be used to associate the nodes of a graph to the nodes of another graph in order to detect the exact or approximate isomorphism between the two graphs.

3.1 The Exact Matching Algorithm

Given any two graphs $\boldsymbol{G}_1$ and $\boldsymbol{G}_2$, we can compute the steady state distributions $\boldsymbol{x}_1^\star(d)$ and $\boldsymbol{x}_2^\star(d)$ for a given value of the parameter d. If the entries in each vector are all distinct, we can match the vertices of the two graphs by constructing the pairs (v_i, v_j), with $v_i \in V_1$ and $v_j \in V_2$, such that $x_1^\star(v_i) = x_2^\star(v_j)$. Because of Proposition 1, the corresponding permutation of the nodes is the only candidate for matching the two graphs. Moreover, if the two vectors do not share the same values, the two graphs are not isomorphic.

Unfortunately, it may happen that some components of $\boldsymbol{x}^\star$ have the same value. A typical case is that of the components corresponding to the vertices without incoming edges (*sources*) which, in fact, have the same value $(1-d)/N$. This effect can be due also to the particular choice of the parameter d, or to symmetries in the graph. When the RW steady state vectors for the two graphs share the same values in the components, but two or more components have the same value, then it is not possible to univocally reconstruct the permutation matrix and all the possible combinations should be tested. However, as showed by the experimental results, this situation rarely happens if the node features are extended by computing the vector $\boldsymbol{x}^\star$ for different values of the parameter d and by considering a reverse topological index, computed as the RW steady state $\boldsymbol{y}^\star$ for the graph obtained from $\boldsymbol{G}$ by reversing the arcs. In particular, this trick overcomes the problem arisen from presence of multiple sources in the graph, which makes the matrix $\boldsymbol{R}$ singular. Moreover, the use of different values for the parameter d allows us to give different weight to the contribution of the layers at a given distance from the current node. As shown in Proposition 2 at most N samples for d are actually meaningful and, consequently, we cannot expect to introduce more discriminant information by adding further values.

Thus, for each vertex i of graph $\boldsymbol{G}$, we compute a real vector

$$\boldsymbol{z}_i^\star(D) = [x_i^\star(d_1), \ldots, x_i^\star(d_n), y_i^\star(d_1), \ldots, y_i^\star(d_n)],$$

being $D = \{d_1, \ldots, d_n\}$ a set of $n \leq N$ distinct values for the parameter d. Thus, the graph $\boldsymbol{G}$ is represented by a matrix $\boldsymbol{Z}^\star(\boldsymbol{G}, D)$, which is composed by N rows and $2 \cdot n$ columns.

The following algorithm naturally arises for the construction of the isomorphism of any two graphs.

Algorithm 1: RWA($\boldsymbol{G}_1, \boldsymbol{G}_2$, N)
begin
s = 0; D = $\emptyset$;
do
  s=s+1; D = D $\bigcup \{d_s\}$;
  $\boldsymbol{Z}^\star(\boldsymbol{G}_1, D) = ComputeSS(\boldsymbol{G}_1, D)$;
  $\boldsymbol{Z}^\star(\boldsymbol{G}_2, D) = ComputeSS(\boldsymbol{G}_2, D)$;
  $P = \text{RebuildPermutation}(\boldsymbol{Z}^\star(\boldsymbol{G}_1, D), \boldsymbol{Z}^\star(\boldsymbol{G}_2, D))$
  if $(P' A_1 P = A_2)$ **return** SuccessfulExit;
until (s < N);
end

The algorithm takes any two graphs $\boldsymbol{G}_1$ and $\boldsymbol{G}_2$ as input. The loop is repeated until the permutation matrix is completely built or the number of samples for d equals the number of nodes N. The function $ComputeSS$ determines the steady-state for the given argument. As proposed in [11], this can be done using a classic iterative scheme, which has been proven [16] to yield any desired precision ϵ with a number of iterations proportional to $\log 1/\epsilon$. The function $RebuildPermutation$ sorts the rows of the two $\boldsymbol{Z}^\star$ matrices and associates the ith row of the sorted $\boldsymbol{Z}^\star(\boldsymbol{G}_1, D)$ with the j7th row of the sorted $\boldsymbol{Z}^\star(\boldsymbol{G}_2, D)$. This association creates pairs of matching vertices in $\boldsymbol{G}_1$ and $\boldsymbol{G}_2$. If all the rows in the two matrices are distinct, then the permutation matrix is univocally obtained. Otherwise, the sampling set D is expanded and the main loop is repeated.

The RW algorithm is polynomial since both the computation of the $\boldsymbol{Z}^\star$ matrices and the sorting step are performed by polynomial algorithms. Notice that the coordinates of the steady-state value $\boldsymbol{x}^\star$, in the worst case, become exponentially close as N grows. However, the number of steps to yield the precision ϵ needed to distinguish those values is only proportional to $\log 1/\epsilon$, which prevents the algorithm from exponential explosion.

Unfortunately, Algorithm 1 might not be able to reconstruct P if, for each sampling set D, the matrices contain colliding rows. In this case, only a part of the mapping can be reconstructed unless all the combinations are tried. However, as shown by the experimental results given in the next section, this circumstance is fortunately rare.

3.2 The Approximate Graph Matching Algorithm

Given any pair of labeled graphs $\boldsymbol{G}_1$ and $\boldsymbol{G}_2$, we can compute the steady states $\boldsymbol{x}_1^\star$ and $\boldsymbol{x}_2^\star$ which summarize both the topological structure and the labels contained in the nodes, using the RW model described in Section 2.2.

Furthermore, we can compute the steady state varying the damping factor d to enrich the node signatures, in order to weigh differently the contribution of the neighborhood of

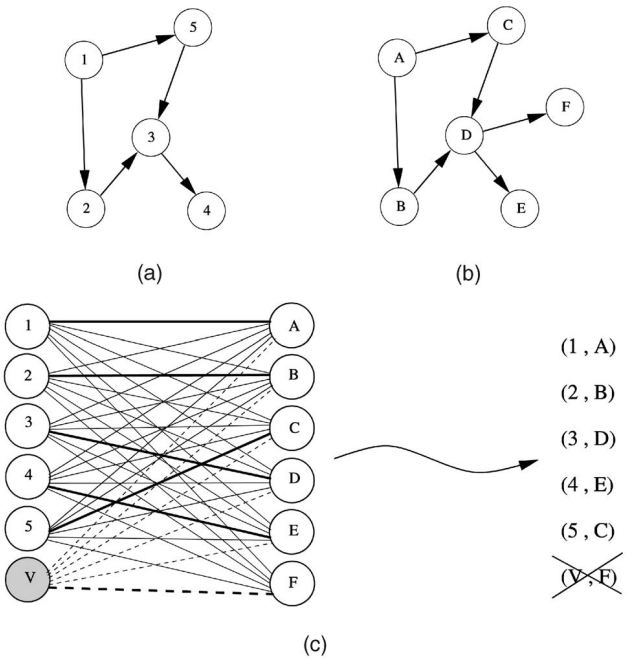

Fig. 2. Two graphs (a) and (b) and the corresponding bipartite graph (c). Since the graphs have a different number of nodes, a virtual vertex V is added. The outcoming edges of the virtual vertex (dotted edges) have a weight set to the maximum cost.

each vertex. Assuming to choose n distinct values of d collected in the sampling vector D, the topological features associated to the ith vertex of a graph can be defined as $\boldsymbol{x}_i^\star(D) = [x_i^\star(d_1), \ldots, x_i^\star(d_n)]$. Finally, the topological features are appended to the node label yielding a signature $LT_i = [L_i, \boldsymbol{x}_i^\star(D)] = [l_{i,1}, \ldots, l_{i,m}, x_i^\star(d_1), \ldots, x_i^\star, (d_n)]$.

In order to determine the approximate match between a pair of given graphs $\boldsymbol{G}_1$ and $\boldsymbol{G}_2$, the signatures attached to each vertex must be compared for creating pairs of matching nodes. This comparison can be performed by solving a bipartite graph matching problem in order to find the optimal matching.

Let V_1 and V_2 be the sets of vertices of graph $\boldsymbol{G}_1$ and $\boldsymbol{G}_2$, respectively. A bipartite graph $\boldsymbol{G}_B = (V_B, E_B)$ can be created, where $V_B = V_1 \cup V_2$ (with $|V_1| = |V_2|$) and $E_B = V_1 \times V_2$. $\boldsymbol{G}_B$ is undirected, and a weight $w(x, y)$ is associated to the edge (x, y) to represent the cost of matching nodes x and y. A matching in $\boldsymbol{G}_B$ is a set of edges $E' \subset E_B$ such that there exists one and only one edge for each node. The cost of the matching is the sum of the weights of the edges in E'. The goal is to find the minimum cost match in $\boldsymbol{G}_B$.

During the creation of $\boldsymbol{G}_B$, the cardinality constraint $|V_1| = |V_2|$ must be satisfied. Such limitation can be practically overcome by adding virtual nodes to the smaller graph with maximum cost associated to their adjacent edges (see Fig. 2).

It is well-known that the optimal matching on a bipartite graph can be determined in polynomial time with different algorithms (see, e.g., [17], [18], [19]). The following algorithm is proposed to find an approximate matching between any two graphs, which relies on the solution of the optimal matching on a bipartite graph.

Algorithm 2: $\texttt{ApproximateRWA}(\boldsymbol{G}_1, \boldsymbol{G}_2, D)$
begin
$\boldsymbol{x}^\star(\boldsymbol{G}_1, D) = \mathrm{ComputeTF}(\boldsymbol{G}_1, D)$;
$\boldsymbol{x}^\star(\boldsymbol{G}_2, D) = \mathrm{ComputeTF}(\boldsymbol{G}_2, D)$;
$\boldsymbol{G}_B = \mathrm{ComputeBipartiteGraph}(\boldsymbol{G}_1, \boldsymbol{G}_2, \boldsymbol{x}^\star(\boldsymbol{G}_1, D),$
$\boldsymbol{x}^\star(\boldsymbol{G}_2, D), \mathrm{weightFunction})$;
$E' = \mathrm{FindOptimalMatch}(\boldsymbol{G}_B)$;
return E';
end

The algorithm takes any two graphs $\boldsymbol{G}_1$ and $\boldsymbol{G}_2$ as input, and the n-dimensional vector D representing the sampling for the d parameter. The number n of the topological features is chosen a priori. The function ComputeTF implements the RW model described in Section 2.2 in order to calculate the steady states $\boldsymbol{x}_1^\star$ and $\boldsymbol{x}_2^\star$ for different values of the damping parameter d. The function ComputeBipartiteGraph takes as input the pair of graphs, the pair of steady state matrices, and a weight function. The topological features collected in the steady state matrices are appended to the node labels, then the bipartite graph $\boldsymbol{G}_B$ is generated, eventually adding virtual nodes in order to met the cardinality constraint. Moreover, a specific weight function to attach the cost to each edge must be specified.[4] Finally, the function FindOptimalMatch computes the optimal match on $\boldsymbol{G}_B$, thus yielding the optimal mapping between the nodes in the two input graphs.

This algorithm can be used for the inexact graph matching problem, for the inexact subgraph matching problem and to determine if a pair of graphs are similar. When $|V_1| \neq |V_2|$ and virtual nodes are added to the bipartite graph, the graph or subgraph match solution is determined considering the first $m = min(|V_1|, |V_2|)$ edges with minimum cost in E'. In fact, the virtual nodes added are guaranteed to have a greater cost. Finally, Algorithm 2 can be applied to compute the similarity of two graphs, defined as the total cost of the optimal matching.

4 Experimental Results

We performed two independent sets of experiment for testing the exact match algorithm on a data set of unlabeled graphs and the inexact match algorithm on a data set of labeled graphs representing images.

4.1 Experiments on Exact Graph Match

Algorithm 1 was evaluated on a subset of the TC-15 graph data set [20]. The TC-15 data set contains 18,200 pairs of isomorphic graphs divided in five classes: randomly connected graphs, bounded valence graphs, irregular bounded valence graphs, regular meshes, and irregular meshes. The subset we chose to perform the experiments contains all the classes except for the regular meshes class.

4. Note that the proposed scheme holds regardless of the chosen weight function, which need not to be defined at this time. In the next section, we will discuss the choice of the Euclidean metrics for experiments of image retrieval.

TABLE 1
Exact Matching Rate on Randomly Connected Graphs from the TC-15 Data Set

Number of Nodes	Arc presence probability		
	0.01	0.05	0.1
20	97.35%	99.70%	100%
40	98.05%	100%	100%
60	98.68%	100%	100%
80	99.19%	100%	100%
100	99.82%	100%	100%
200	99.99%	100%	100%
400	100%	100%	100%
600	100%	100%	100%
800	100%	100%	100%
1000	100%	100%	100%

The RW Algorithm has been evaluated in order to determine how the topological regularity and the dimension of the graphs affect the exact matching reconstruction. The results are evaluated using an "Exact Matching Rate" that represents the percentage of rows and columns of the permutation matrix P which were reconstructed correctly. Since, in [21], the performance of five different exact matching algorithms are compared with respect to their execution time, showing that the VF algorithm outperforms all other methods on the TC-15 data set, we have also compared the execution time of the RW algorithm with respect to the VF algorithm.[5]

4.1.1 Randomly Connected Graphs

In this set of graphs, the arcs connect vertices without any structural regularity. These graphs are generated by choosing a value η which represents the probability that an arc is present between two distinct vertices. The arcs are added until the desired arc density is reached. If the resulting graph is not connected, appropriate arcs are added to generate a connected graph. The class collects 3,000 pairs of graphs generated using $\eta = 0.01, 0.05, 0.1$. The size of the graphs ranges from 20 to 1,000 vertices.

The achieved results are reported in Table 1, and show that the exact matching rate increases both with the number of vertices and with the arc density. The number of samples for the parameter d needed to reconstruct the exact matching is just one (only one iteration of the main loop of the algorithm) if the arc density is $\eta = 0.1$ or $\eta = 0.05$ and if the number of vertices is greater than 100. Thus, only with very sparse graphs with a small number of nodes the sampling of the d parameter is actually needed. The comparison of the execution times for the two algorithms, with respect to the graph size and arc density, is reported in Fig. 3. The execution time of the RW algorithm is always significantly better than the VF algorithm. When applied on sparse graphs with a small number of nodes, the RW algorithm requires to use more values for the parameter d to reconstruct the exact matching. However, the very efficient way in which both the RW steady state computation and the reconstruction of the matrix P are implemented allow us to find the exact match very fast. Furthermore, the relative performance increases with the size of the graphs.

5. We have implemented the VF algorithm using the VFLIB (available at http://amalfi.dis.unina.it/graph/).

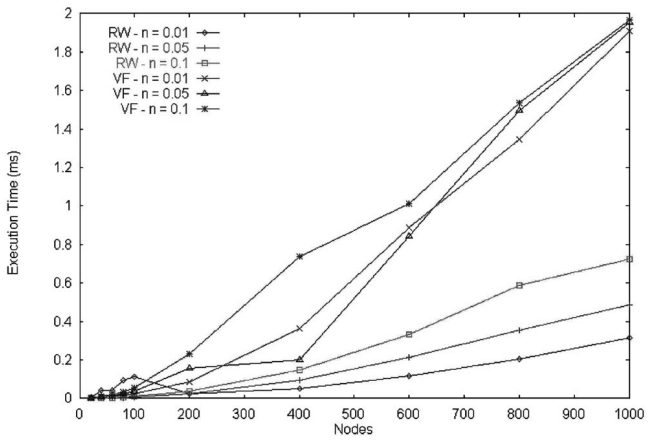

Fig. 3. Execution time comparison between the RW algorithm and VF algorithm on Randomly connected graphs.

4.1.2 Bounded Valence Graphs and Irregular Bounded Valence Graphs

Graphs belonging to the Bounded Valence Graphs (BVGs) class have a fixed valence, and are generated adding random arcs until the valence of all the nodes does not exceed a predefined value. On the contrary, in the Irregular bounded valence graphs (Irregular BVGs), the average valence of the nodes is still bounded, but the single nodes can have a valence which is not bounded by a predefined value. An Irregular BVGs is generated taking as input a BVG and moving a certain number of arcs from the nodes they are attached to, to other nodes. In particular, the 10 percent of all the arcs are moved, in order to introduce some irregularity in the original BVG. Both the classes collect 3,000 pairs of isomorphic graphs and were generated using a valence of 3, 6, and 9. As in the randomly connected graphs class, the size of the graphs ranged from 20 to 1,000.

The results obtained by the RW Algorithm are presented in Table 2. The results show that exact matching rate increases with the valence of the graphs. Moreover, the irregularity added in the Irregular BVGs seems to not strongly affect the behavior of the algorithm. The number of samples for the parameter d, needed to reconstruct the permutation matrix, is just one if the valence is 6 or 9. Instead, if the valence is 3, the algorithm requires to use more values for the parameter d (about 5 distinct values, on average), to achieve the best exact matching rate.

TABLE 2
Exact Matching Rate on BVGs and Irregular BVGs Belonging to the TC-15 Data Set

Number of Nodes	Valence					
	03	Irregular 03	06	Irregular 06	09	Irregular 09
20	99.25%	99.70%	100%	100%	100%	100%
40	99.18%	98.75%	100%	100%	100%	100%
60	99.35%	97.88%	100%	100%	100%	100%
80	97.89%	100%	100%	100%	100%	100%
100	96.72%	99.83%	100%	100%	100%	100%
200	97.25%	98.35%	100%	100%	100%	100%
400	98.25%	99.22%	100%	100%	100%	100%
600	99.86 %	99.35%	100%	100%	100%	100%
800	99.25%	98.55%	100%	100%	100%	100%
1000	98.35%	99.14%	100%	100%	100%	100%

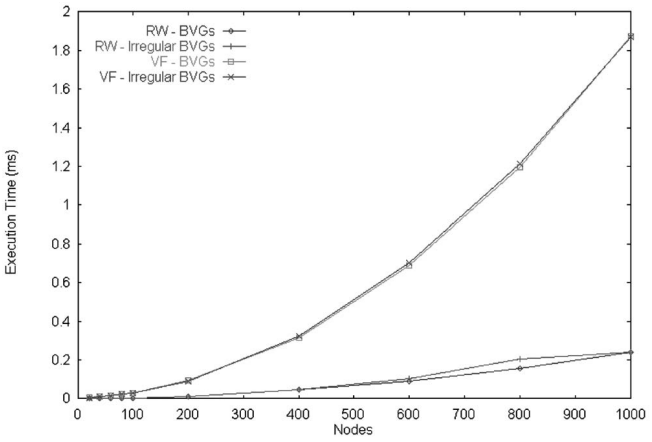

Fig. 4. Execution time comparison between the RW and the VF algorithms on BVGs and Irregular BVGs with a valence equal to 6.

The comparison of the execution time of the RW and VF algorithms, referred to graphs having a valence equal to 6, is reported in Fig. 4. The execution time of the RW algorithm is better than the VF algorithm, particularly when the size of the graphs increases.

4.1.3 Irregular Meshes

The Irregular meshes collected in this class are generated adding random arcs, uniformly distributed, to regular meshes. The number of added arcs is rN, where r is an irregularity density value and N is the number of nodes of the regular mesh. Notice that, the smaller is r, the more symmetric are the generated Irregular meshes. The Irregular meshes belonging to the TC-15 data set are further divided into three distinct classes: 2D, 3D, and 4D meshes. Each class collects Irregular meshes generated using $r = 0.2, 0.4, 0.6$. Finally, the TC-15 data set contains 3,000 pairs of isomorphic 2D meshes (from 16 to 1,024 nodes), 2,400 pairs of isomorphic 3D meshes (from 27 to 1,000 nodes), and 1,500 pairs of 4D meshes (from 16 to 1,296 nodes). The results obtained by the RW algorithm processing Irregular meshes are reported in Table 3. Also in this case, the permutation matrix is completely determined for each kind of irregular mesh except for very small graphs. RW algorithm is able to reconstruct the permutation matrix also processing the less irregular meshes ($r = 0.2$). Moreover, the number of samples of d needed to reconstruct the exact matching is just one, and no improvement of the exact matching rate is achieved considering more values of d. Finally, in Fig. 5, the comparison of the execution time of the RW algorithm and the VF algorithm is reported. The RW algorithm outperforms the VF algorithm also on the Irregular meshes class.

TABLE 3
Exact Matching Rate on Irregular Meshes Belonging to the TC-15 Data Set

Kind of Mesh	Number of Nodes	Irregularity density value		
		0.2	0.4	0.6
2D Mesh	16	99.18%	100%	100%
	36	100%	100%	100%
	64	100%	100%	100%
	81	100%	100%	100%
	100	100%	100%	100%
	196	100%	100%	100%
	400	100%	100%	100%
	576	100%	100%	100%
	784	100%	100%	100%
	1024	100%	100%	100%
3D Mesh	27	100%	100%	100%
	64	100%	100%	100%
	125	100%	100%	100%
	216	100%	100%	100%
	343	100%	100%	100%
	512	100%	100%	100%
	729	100%	100%	100%
	1000	100%	100%	100%
4D Mesh	16	95.38%	95.28%	99.88%
	81	100%	100%	100%
	256	100%	100%	100%
	625	100%	100%	100%
	1296	100%	100%	100%

4.2 Experiments on Inexact Graph Matching

In order to evaluate the effectiveness of Algorithm 2, some experiments on an image retrieval task were performed.

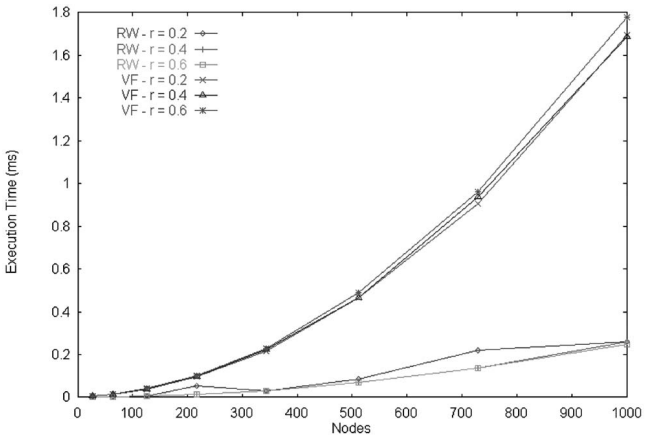

Fig. 5. Execution time comparison between the RW algorithm and VF algorithm on 3D Irregular meshes.

The image data set used in the experiments is the Columbia Object Image Library (COIL-100 database—see Fig. 6) [22]. The COIL-100 contains 7,200 color images of 100 objects (72 images per object, one at every 5 degrees of rotation—see Fig. 7) and it has been used in the past to evaluate the performances of 3D object recognition systems [23], [24]. The object appearance varies both for its geometric and reflectance characteristics.

Each image is represented by a Region Adjacency Graph (RAG). The nodes in the graph correspond to the connected homogeneous regions in the image obtained using a segmentation algorithm [25]. Each node is labeled with a vector of features which describes visual properties of the corresponding region in the image (area, perimeter, average color, minimum bounding box coordinates). An edge between a pair of nodes denotes the adjacency of the two regions associated to the nodes. Thus, the graph topology codes the spatial relationships among the regions in the image. In particular, two connected regions R_1, R_2 are *adjacent* if, for each pixel $a \in R_1$ and $b \in R_2$, there exists a path connecting a and b, entirely lying into $R_1 \cup R_2$.

Each COIL-100 image was segmented and subsequently represented by a RAG composed by an average of 30 nodes.

The performance of the inexact matching algorithm was evaluated in an image retrieval task. Given a query image I, the system retrieves the M most similar images from the image database. Algorithm 2 can be easily adapted to find the best match of a graph in a database. Given the query graph $\boldsymbol{G}_Q$, $N-1$ distinct bipartite graphs are generated to compute the cost for matching the graph $\boldsymbol{G}_Q$ with respect to the other graphs in the database. The $N-1$ bipartite graph matching problems were solved using the Hungarian method [17] and the total cost of each optimal match was computed. Thus, the optimal matches were sorted in ascending order with respect to their total cost. The first M matches in the ordered list corresponded to the M graphs which were the answer to the query $\boldsymbol{G}_Q$.

In order to evaluate the retrieval performance on the COIL-100 data set, we defined the set of admissible results for each query image. An image in the result set is a hit if it represents the same object of the query image. However, since each 3D object is represented with images acquired all around, it may happen that different views of the same object acquired at various degrees of rotation can be very different from one another. Thus, given I and its degree of

Fig. 6. Images of the 100 objects of the COIL database. (All trademarks remain the property of their respective owners. All trademarks and registered trademarks are used strictly for educational and scholarly purposes and without intent to infringe on the mark owners.)

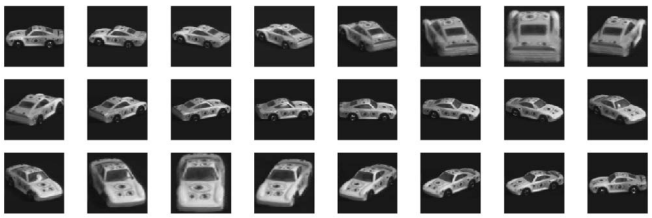

Fig. 7. Twenty-four of the 72 images of a COIL object.

TABLE 4
Accuracy Rate Obtained Varying the Number of Clusters in the K-Means Algorithm

Number of clusters	Accuracy Rate
10	0.9412
20	0.9735
40	0.952
60	0.9504
80	0.9299

The system retrieves four similar images and computes the topological features using five distinct damping factors.

rotation α_I, we consider as similar the images of the same object acquired within a degree of rotation in $\alpha_I \pm \beta$. For the results reported in this paper, we chose $\beta = 35°$. The performance was evaluated by an *Accuracy Rate*, that represents the number of result images which satisfy these criteria divided by the total number of results.

In the experiments, we applied both the tendency function method and the estimation based on clustering to determine the transition probabilities of the RW model as described in Section 2.2. Furthermore, also the influence of the number of distinct samples of the damping parameter d was investigated. The results are also reported varying the number of retrieved images. The accuracy rate reported in the following tables was computed as the average accuracy rate obtained by choosing each image as the query.

The tendency functions were devised in order to yield a high probability of following arcs linking vertices with similar labels and of jumping to vertices which correspond to large regions. Two adjacent vertices were considered similar if the corresponding regions have similar colors. Thus, the tendency functions were chosen as $f_l(L_i, L_j) = \frac{1}{\|RGB_i - RGB_j\|}$ and $f_j(L_i, L_j) = f_j(L_i) = Area(i)$, where RGB_i represents the color vector of the ith region in the RGB color space and $Area(i)$ represents the number of pixels of the ith region. Notice that the tendency function f_j depends only on the target of the jump.

When applying the RW model with estimated transition probabilities, the COIL-100 data set was split into a training set and a test set. The training and the test sets contain 1,400 and 5,800 images, respectively. The training set is composed by 14 images per object, acquired every $25°$ of rotation; the other images form the test set. Then, the labels attached to the nodes of the graphs in the training set were clusterized using the K-means algorithm. The number of clusters needed to optimally represent the feature space was evaluated empirically (see Table 4). In order to implement the RW described in Section 2.2, the distribution $prob((q,p)|c_q, c_p)$ is estimated by counting the number of arcs which link vertices in c_q to vertices in c_p.

TABLE 5
Accuracy Rate Obtained Using Both the Visual and Topological Features or Only the Visual Features

Retrieved Images	Visual and Topological Features	Visual Features
4	0.9735	0.8121
9	0.9544	0.8004
14	0.9337	0.7312

The transition probabilities of the RW model are computed using clustering (20 clusters).

TABLE 6
Comparison of the Transition Probabilities Estimation Methods

Retrieved Images	Tendency Functions	Clustering
4	0.9707	0.9735
9	0.9561	0.9544
14	0.9373	0.9337

The method based on tendency functions and the one based on estimation with label clustering (20 clusters) seem to be equivalent.

Furthermore, the cost associated to each edge in the bipartite graph used to compute the optimal mapping among the nodes was set to the weighted Euclidean distance between the signatures of the nodes. In particular, the contribution of the visual features and that of the topological features were normalized by the number of components in each group.

An experiment was performed to evaluate the contribution of the topological features. This comparison was performed by considering both the visual and topological features or only the visual features. The results (see Table 5) show that the topological features improve significantly the retrieval accuracy validating their use.

As shown in Table 6, the performance of the algorithm is not significantly affected by the computation method for the transition probabilities.

Finally, the influence of the number of distinct samples for d has been evaluated using 5, 10, and 15 values. As shown in Table 7, the results do not seem to be affected by this choice. The table summarizes also the retrieval results obtained on the test set of COIL-100.

TABLE 7
Effect of the Number of Samples for the Damping Factor d Chosen to Compute the Topological Features

Retrieved Images	Number of distinct d		
	5	10	15
4	0.9735	0.9544	0.9337
9	0.9544	0.9508	0.9509
14	0.9337	0.9332	0.9332

The reported results are obtained using the estimation of the RW parameters using clustering (K = 20 clusters). The best accuracy rate on the test set of COIL-100 (more than 97 percent) is obtained with five samples for d and retrieving four images.

5 Conclusions

In this paper, we have introduced a general framework for graph matching which is based on the novel notion of Markovian graph spectral analysis. The theory is related to classic spectral analysis, but it involves directly the nodes and, therefore, the notion of spectrum which is introduced turns out to be very well-suited for different problems of graph matching. In particular, a polynomial algorithm is given for solving the graph isomorphism problem, under the restriction of dealing with Markovian spectrally distinguishable graphs, a class of graphs which does not seem to be easily reducible to others proposed in the literature. The theory is also shown to be effective for partial and approximate graph matching when used in conjunction with bipartite graph matching algorithms. Successful experimental results are given on the TC-15 graph database for the graph isomorphism problem and on the COIL-100 visual database, where the visual retrieval is based on partial and approximate graph matching.

Finally, it is worth mentioning that the general framework proposed in this paper can be seen in conjunction with learning theories aimed at forcing the value of the probability of the random walk on some given nodes of the graph (see, e.g., [26]). This further possibility is likely to enrich the notion of Markovian graph matching and to open to the doors also to the solution of complex problems of subgraph matching that are, for instance, of crucial importance in image retrieval.

References

[1] J.R. Ullmann, "An Algorithm For Subgraph Isomorphism," *J. ACM,* vol. 23, no. 1, pp. 31-42, 1976.

[2] D.C. Schmidt and L.E. Druffel, "A Fast Backtracking Algorithm to Test Directed Graphs For Isomorphism Using Distance Matrices," *J. ACM,* vol. 23, no. 3, pp. 433-445, 1976.

[3] B.D. McKay, "Practical Graph Isomorphism," *Congressus Numerantium,* vol. 30, pp. 45-87, 1981.

[4] L.P. Cordella, P. Foggia, C. Sansone, and M. Vento, "Performance Evaluation of the VF Graph Matching Algorithm," *Proc. 10th Int'l Conf. Image Analysis and Processing,* pp. 1172-1177, 1999.

[5] M.A. Eschera and K.S. Fu, "A Graph Distance Measure for Image Analysis," *IEEE Trans. Systems, Man, and Cybernetics,* vol. 14, pp. 398-407, 1984.

[6] W.J. Christmas, J. Kittler, and M. Petrou, "Structural Matching in Computer Vision Using Probabilistic Relaxation," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 17, no. 8, pp. 749-764, Aug. 1995.

[7] K.A. DeJong and W.M. Spears, "Using Genetic Algorithm to Solve NP-Complete Problems," *Proc. Int'l Conf. Genetic Algorithms,* pp. 124-132, 1989.

[8] S. Umeyama, "An Eigendecomposition Approach to Weighted Graph Matching Problems," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 10, no. 5, pp. 695-703, May 1988.

[9] B. Luo and E.R. Hancock, "Structural Graph Matching Using the EM Algorithm and Singular Value Decomposition," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 23, no. 10, pp. 1120-1136, Oct. 2001.

[10] F. DePiero, M. Trivedi, and S. Serbin, "Graph Matching Using a Direct Classification of Node Attendance," *Pattern Recognition,* vol. 29, no. 6, pp. 1031-1048, 1996.

[11] L. Page, S. Brin, R. Motwani, and T. Winograd, "The PageRank Citation Ranking: Bringing Order to the Web," technical report, Computer Science Dept., Stanford Univ., 1999.

[12] M. Diligenti, M. Gori, and M. Maggini, "A Unified Probabilistic Framework for Web Page Scoring Systems," *IEEE Trans. Knowledge and Data Eng.,* vol. 16, no. 1, pp. 4-16, Jan. 2004.

[13] E. Seneta, *Non-Negative Matrices and Markov Chains.* Springer-Verlag, 1981.

[14] L. Padulo and M. Arbib, *System Theory—A Unified Approach to Continous and Discrete Systems.* Philadelphia, London, Toronto: W.B. Saunders Company, 1974.

[15] A. Pasini, "On Two Matrices Related to the Transition Matrix of a Graph," technical report, Dipartimento di Matematica, Univ. di Siena, 2004.

[16] M. Bianchini, M. Gori, and F. Scarselli, "Inside PageRank," *ACM Trans. Internet Technology,* 2005.

[17] H.W. Kuhn, "The Hungarian Method for the Assignment Problem," *Naval Research Logistics Quarterly,* vol. 2, pp. 83-97, 1955.

[18] P.K. Agarwal and K. Varadarajan, "Approximation Algorithms for Bipartite and Non-Bipartite Matching in the Plane," *Proc. 10th Ann. ACM-SIAM Symp. Discrete Algorithms,* pp. 805-814, 1999.

[19] H.N. Gabow and R.E. Tarjan, "Faster Scaling Algorithms for Network Problems," *SIAM J. Computing,* vol. 18, no. 5, pp. 1013-1036, 1989.

[20] P. Foggia, C. Sansone, and M. Vento, "A Database of Graphs for Isomorphism and Subgraph Isomorphism Benchmarking," *Proc. Third IAPR TC-15 Int'l Workshop Graph-Based Representations in Pattern Recognition,* pp. 176-187, 2001.

[21] P. Foggia, C. Sansone, and M. Vento, "A Performance Comparison of Five Algorithms for Graph Isomorphism," *Proc. Third IAPR TC-15 Int'l Workshop Graph-Based Representations in Pattern Recognition,* pp. 188-199, 2001.

[22] S.A. Nene, S.K. Nayar, and H. Murase, "Columbia Object Image Library (COIL-100)," technical report, Columbia Univ., 1996.

[23] S.A. Nene, S.K. Nayar, and H. Murase, "Real-Time 100 Object Recognition System," *Proc. IEEE Int'l Conf. Robotics and Automation,* pp. 2321-2325, 1996.

[24] M. Pontil and A. Verri, "Support Vector Machines for 3D Object Recognition," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 20, no. 6, pp. 637-646, June 1998.

[25] C. DeMauro, M. Diligenti, M. Gori, and M. Maggini, "Similarity Learning for Graph-Based Image Representations," *Proc. Third IAPR TC-15 Int'l Workshop Graph-Based Representations in Pattern Recognition,* pp. 250-259, 2001.

[26] M. Diligenti, M. Gori, and M. Maggini, "A Learning Algorithm for Web Page Scoring Systems," *Proc. 18th Int'l Joint Conf. Artificial Intelligence,* pp. 575-580, 2003.

Marco Gori received the Laurea in electronic engineering from Università di Firenze, Italy, in 1984, and the PhD degree in 1990 from Università di Bologna, Italy. From October 1988 to June 1989, he was a visiting student at the School of Computer Science (McGill University, Montreal). In 1992, he became an associate professor of computer science at Università di Firenze and, in November 1995, he joined the University of Siena, where he is currently full professor. His main research interests are in neural networks, pattern recognition, and applications of machine learning to information retrieval on the Internet. He has leaded a number of research projects on these themes with either national or international partners, and has been involved in the organization of many scientific events. Dr. Gori serves (served) as an associate editor of a number of technical journals related to his areas of expertise, including *Pattern Recognition*, the *IEEE Transactions Neural Networks*, *Neurocomputing*, and the *International Journal on Pattern Recognition and Artificial Intelligence*. He is the Italian chairman of the IEEE Computational Intelligence Society and is the president of the Italian Association for Artificial Intelligence. He is also a fellow of the IEEE.

Marco Maggini received the Laurea degree cum laude in electronic engineering from the University of Firenze in February 1991 and the PhD in computer science and control systems in 1995 from the University of Firenze. In February 1996, he became an assistant professor of computer engineering at the School of Engineering of the University of Siena, where, since March 2001, he has been an associate professor. His main research interests are machine learning, neural networks, human-machine interaction, technologies for distributing and searching information on the Internet, and nonstructured databases. He has been collaborating with the NEC Research Institute, Princeton, New Jersey, on parallel processing, neural networks, and financial time series prediction. He is an associate editor of the *ACM Transactions on Internet Technology*. He has been a guest editor of a special issue of the ACM TOIT on machine learning for the Internet. He contributed to the organization of international and national scientific events. He is member of the IAPR-IC and the IEEE Computer Society.

Lorenzo Sarti received the Laurea degree in computer since in 2001 from the University of Florence, Italy. Currently, he is a PhD student in information engineering at the University of Siena, Italy. His main research interests are pattern recognition in structured domains, machine learning techniques applied to image analysis and understanding, visual databases, and visual information retrieval.