A path following algorithm for the graph matching problem

Mikhail Zaslavskiy, Francis Bach, Jean-Philippe Vert

▶ To cite this version:

A path following algorithm for the graph matching problem

Mikhail Zaslavskiy, Francis Bach, and Jean-Philippe Vert [*][†][‡][§]

February 2, 2008

Abstract

We propose a convex-concave programming approach for the labeled weighted graph matching problem. The convex-concave programming formulation is obtained by rewriting the weighted graph matching problem as a least-square problem on the set of permutation matrices and relaxing it to two different optimization problems: a quadratic convex and a quadratic concave optimization problem on the set of doubly stochastic matrices. The concave relaxation has the same global minimum as the initial graph matching problem, but the search for its global minimum is also a hard combinatorial problem. We therefore construct an approximation of the concave problem solution by following a solution path of a convex-concave problem obtained by linear interpolation of the convex and concave formulations, starting from the convex relaxation. This method allows to easily integrate the information on graph label similarities into the optimization problem, and therefore to perform labeled weighted graph matching. The algorithm is compared with some of the best performing graph matching methods on four datasets: simulated graphs, QAPLib, retina vessel images and handwritten chinese characters. In all cases, the results are competitive with the state-of-the-art.

Keywords: graph matching, graph algorithms, convex programming, gradient methods, machine learning, classification, image processing

1 Introduction

The graph matching problem is among the most important challenges of graph processing, and plays a central role in various fields of pattern recognition. Roughly speaking, the problem consists in finding a correspondence between vertices of two given graphs which is optimal in some sense. Usually, the optimality refers to the alignment of graph structures and, when available, of vertices labels, although other criteria are possible as well. A non-exhaustive list of graph matching applications includes document processing tasks like optical character recognition [LL99, AAI95], image analysis (2D and 3D) [WH05, BR00, CR02, CC05], or bioinformatics [RJB07, WMFH04, Tay02].

---

[*]Mikhail Zaslavskiy is with the Centre for Computational Biology and the Centre for Mathematical Morphology; Jean-Philippe Vert is with the Centre for Computational Biology, École des Mines de Paris, 35 rue Saint-Honoré, 77305 Fontainebleau, France. They are also with the Institut Curie, Section Recherche, and with INSERM U900.

[†]E-mail: mikhail.zaslavskiy@ensmp.fr, jean-philippe.vert@ensmp.fr

[‡]Francis Bach is with INRIA-Willow Project, École Normale Supérieure, 45 rue d'Ulm, 75230 Paris, France

[§]E-mail: francis.bach@mines.org

During the last decades, many different algorithms for graph matching have been proposed. Because of the combinatorial nature of this problem, it is very hard to solve it exactly for large graphs, however some methods based on incomplete enumeration may be applied to search for an exact optimal solution in the case of small or sparse graphs. Some examples of such algorithms may be found in [SD76, Ull76, CFSV91].

Another group of methods includes approximate algorithms which are supposed to be much more scalable. The price to pay for the scalability is that the solution found is usually only an approximation of the optimal matching. Approximate methods may be divided into two groups of algorithms. The first group is composed of methods which use spectral representations of adjacency matrices, or equivalently embed the vertices into a Euclidean space where linear or nonlinear matching algorithms can be deployed. This approach was pioneered by Umeyama [Ume88], while further different methods based on spectral representations were proposed in [SB92, CR02, BR00, WH05, CK04]. The second group of approximate algorithms is composed of algorithms which work directly with graph adjacency matrices, and typically involve a relaxation of the complex discrete optimization problem. The most effective algorithms were proposed in [AS93, SA96, DTC01, CC05].

An interesting instance of the graph matching problem is the matching of labeled graphs. In that case graph vertices have associated labels, which may be numbers, categorical variables, etc... The important point is that there is also a similarity measure between these labels. Therefore, when we search for the optimal correspondence between vertices, we search a correspondence which matches not only the structures of the graphs but also vertices with similar labels. Some widely used approaches for this application only use the information about similarities between graph labels. In vision, one such algorithm is the shape context algorithm proposed in [BMP02], which involves a very efficient algorithm of node label construction. Another example is the BLAST-based algorithms in bioinformatics such as the Inparanoid algorithm [KME05], where correspondence between different protein networks is established on the basis of BLAST scores between pairs of proteins. The main advantages of all these methods are their speed and simplicity. However the main drawback of these methods is that they do not take into account information about the graph structure. Some graph matching methods try to combine information on graph structures and vertex similarities, examples of such method are presented in [DTC01, RJB07].

In this article we propose an approximate method for labeled weighted graph matching, based on a convex-concave programming approach which can be applied for matching of large size graphs. Our method is based on a formulation of the labeled weighted graph matching problem as a quadratic assignment problem (QAP) over the set of permutation matrices, where the quadratic term encodes the structural compatibility and the linear term encodes local compatibilities. We propose two relaxations of this problem, resulting in one quadratic convex and one quadratic concave optimization problem on the set of doubly stochastic matrices. While the concave relaxation has the same global minimum as the initial QAP, solving it is also a hard combinatorial problem. We find a local minimum of this problem by following a solution path of a family of convex-concave optimization problems, obtained by linearly interpolating the convex and concave relaxations. Starting from the convex formulation with a unique local (and global) minimum, the solution path leads to a local optimum of the concave relaxation. We refer to this procedure as the PATH algorithm. We perform an extensive comparison of this PATH algorithm with several state-of-the-art matching methods on small simulated graphs and various QAP benchmarks, and show that it consistently provides state-of-the-art performances while scaling to graphs of up to a few thousands vertices on a modern desktop computer. We further illustrate the use of the algorithm on two applications in image processing, namely the matching of retina images based on vessel organization, and the matching of handwritten chinese characters.

The rest of the paper is organized as follows: Section 2 presents the mathematical formulation of the graph matching problem. In Section 3 we present our new approach. Then, in Section 4, we present the comparison of our method with Umeyama's algorithm and the linear programming approach on the example of artificially simulated graphs. In Section 5, we test our algorithm on the QAP benchmark library, and we compare obtained results with the results published in [DTC01] for the QBP algorithm and graduated assignment algorithms. Finally, in Section 6 we present two examples of real-world image processing.

2   Problem description

A graph $G = (V, E)$ of size N is defined by a finite set of vertices $V = \{1, \ldots, N\}$ and a set of edges $E \subset V \times V$. We consider only undirected graphs with no self-loop, i.e., such that $(i, j) \in E \implies (j, i) \in E$ and $(i, i) \notin E$ for any vertices $i, j \in V$. Each such graph can be equivalently represented by a symmetric adjacency matrix A of size $|V| \times |V|$, where A_{ij} is equal to one if there is an edge between vertex i and vertex j, zero otherwise. An interesting generalization is a weighted graph which is defined by association of real values w_{ij} (weights) to all edges of graph G. Such graphs are represented by real valued adjacency matrices A with $A_{ij} = w_{ij}$. This generalization is important because in many applications the graphs of interest have associated weights for all their edges, and taking into account these weights may be crucial in construction of efficient methods. In the following when we talk about "adjacency matrix" we mean real-valued "weighted" adjacency matrix.

Given two graphs G and H with the same number of vertices N, the problem of matching G and H consists in finding a correspondence between vertices of G and vertices of H which aligns G and H in some optimal way. We will consider in Section 3.8 an extension of the problem to graphs of different sizes. For graphs with the same size N, the correspondence between vertices is a permutation of N vertices, which can be defined by a permutation matrix P, i.e., a $\{0, 1\}$-valued $N \times N$ matrix with exactly one entry 1 in each column and each row. The matrix P entirely defines the mapping between vertices of G and vertices of H, P_{ij} being equal to 1 if the i-th vertex of G is matched to the j-th vertex of H, and 0 otherwise. After applying the permutation defined by P to the vertices of H we obtain a new graph isomorphic to H which we denote by $P(H)$. The adjacency matrix of the permuted graph, $A_{P(H)}$, is simply obtained from A_H by the equality $A_{P(H)} = P A_H P^T$.

In order to assess whether a permutation P defines a good matching between the vertices of G and those of H, a quality criterion must be defined. Although other choices are possible, we focus in this paper on measuring the discrepancy between the graphs after matching, by the number of edges (in the case of weighted graphs, it will be the total weight of edges) which are present in one graph and not in the other. In terms of adjacency matrices, this number can be computed as:

$$F_0(P) = ||A_G - A_{P(H)}||_F^2 = ||A_G - P A_H P^T||_F^2, \tag{1}$$

where $||.||_F$ is the Frobenius matrix norm defined by $||A||_F^2 = \mathrm{tr} A^T A = (\sum_i \sum_j A_{ij}^2)$. A popular alternative to the Frobenius norm formulation (1) is the 1-norm formulation obtained by replacing the Frobenius norm by the 1-norm $||A||_1 = \sum_i \sum_j |A_{ij}|$, which is equal to the Frobenius norm when comparing $\{0, 1\}$-valued matrices, but may differ in the case of general matrices.

Therefore, the problem of graph matching can be reformulated as the problem of minimizing $F_0(P)$ over the set of permutation matrices. This problem has a combinatorial nature and there

is no known polynomial algorithm to solve it [GM79]. It is therefore very hard to solve it in the case of large graphs, and numerous approximate methods have been developed.

An interesting generalization of the graph matching problem is the problem of *labeled* graph matching. Here each graph has associated labels to all its vertices and the objective is to find an alignment that fits well graph labels and graph structures at the same time. If we let C_{ij} denote the cost of fitness between the i-th vertex of G and the j-th vertex of H then the matching problem based only on label comparison can be formulated as follows:

$$\min_{P \in \mathcal{P}} \ \mathrm{tr}(C^T P) = \sum_{i=1}^{N} \sum_{j=1}^{N} C_{ij} P_{ij} = \sum_{i=1}^{N} C_{i,P(i)}, \tag{2}$$

where $\mathcal{P}$ denotes the set of permutation matrices. A natural way of unifying of (2) and (1) to match both the graph structure and the vertice's labales is then to minimize a convex combination [DTC01]:

$$\min_{P \in \mathcal{P}} \ (1 - \alpha) F_0(P) + \alpha \mathrm{tr}(C^T P), \tag{3}$$

that makes explicit, through the parameter $\alpha \in [0, 1]$, the trade-off between cost of individual matchings and faithfulness to the graph structure. A small α value emphasizes the matching of structures, while a large α value gives more importance to the matching of labels.

2.1   Permutation matrices

Before describing how we propose to solve (1) and (3), we first introduce some notations and bring to notice some important characteristics of these optimization problems. They are defined on the set of permutation matrices, which we denoted by $\mathcal{P}$. The set $\mathcal{P}$ is a set of extreme points of the set of doubly stochastic matrices $\mathcal{D} = \{A : A1_N = 1_N, \ A^T 1_N = 1_N, \ A \geq 0\}$, where 1_N denotes the N-dimensional vector of all ones [BL00]. This shows that when a linear function is minimized over the set of doubly stochastic matrices $\mathcal{D}$, a solution can always be found in the set of permutation matrices. Consequently, minimizing a linear function over $\mathcal{P}$ is in fact equivalent to a linear program and can thus be solved in polynomial time by, e.g., interior point methods [BV03]. In fact, one of the most efficient methods to solve this problem is the Hungarian algorithm, which uses another strategy to solve this problem in $O(N^3)$.

At the same time $\mathcal{P}$ may be considered as a subset of orthonormal matrices $\mathcal{O} = \{A : A^T A = I\}$ of $\mathcal{D}$ and in fact $\mathcal{P} = \mathcal{D} \cap \mathcal{O}$. An (idealized) illustration of these sets is presented in Figure 1.

2.2   Approximate methods: existing works

A good review of graph matching algorithms may be found in [DPCM04]. Here we only present a brief description of some approximate methods which illustrate well ideas behind two subgroups of these algorithms. As mentioned in the introduction, one popular approach to find approximate solutions to the graph matching problem is based on the spectral decomposition of the adjacency matrices of the graphs to be matched. In this approach, the singular value decompositions of the graph adjacency matrices are used:

$$A_G = U_G \Lambda_G U_G^T, \ A_H = U_H \Lambda_H U_H^T, \tag{4}$$

where the columns of the orthogonal matrices U_G and U_H consist of eigenvectors of A_G and A_H respectively, and Λ_G and Λ_H are diagonal matrices of eigenvalues.
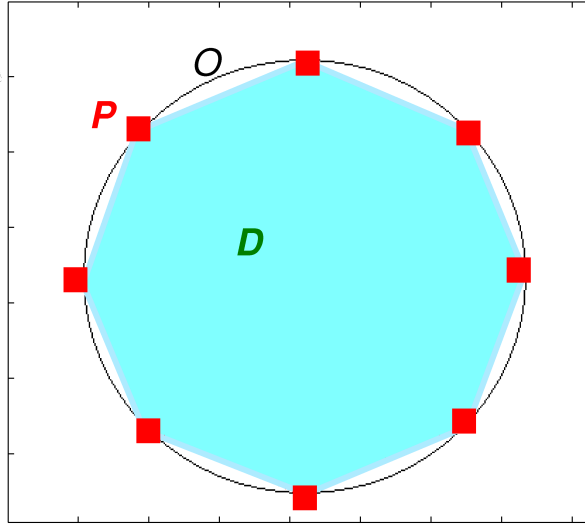
Figure 1: Relation between three matrix sets. $\mathcal{O}$—set of orthogonal matrices, $\mathcal{D}$ — set of doubly stochastic matrices, $\mathcal{P} = D \cap O$—set of permutation matrices.

If we consider the rows of eigenvector matrices U_G and U_H as graph node coordinates in eigenspaces, then we can match the vertices with similar coordinates through a variety of methods [Ume88, CR02, CK04]. These methods suffer from the fact that the spectral embedding of graph vertices is not uniquely defined. First, the unit norm eigenvectors are at most defined up to a sign flip and we have to choose their signs synchronously. Although it is possible to use some normalization convention, such as choosing the sign of each eigenvector in such a way that the biggest component is always positive, this usually does not guarantee a perfect sign synchronization, in particular in the presence of noise. Second, if the adjacency matrix has multiple eigenvalues, then the choice of eigenvectors becomes arbitrary within the corresponding eigen-subspace, as they are defined only up to rotations [GL96].

One of the first spectral approximate algorithms was presented by Umeyama [Ume88]. To avoid the ambiguity of eigenvector selection, Umeyama proposed to consider the absolute values of eigenvectors. According to this approach, the correspondence between graph nodes is established by matching the rows of $|U_G|$ and $|U_H|$ (which are defined as matrices of absolute values). The criterion of optimal matching is the total distance between matched rows, leading to the optimization problem:

$$\min_{P \in \mathcal{P}} \| \, |U_G| - P|U_H| \, \|_F \, ,$$

or equivalently:

$$\max_{P \in \mathcal{P}} \text{tr}(|U_H||U_G|^{\text{T}}P) \, . \tag{5}$$

The optimization problem (5) is a linear program on the set of permutation matrices which can be solved by the Hungarian algorithm in $O(N^3)$ [McG83, Kuh55].

The second group of approximate methods consists of algorithms which work directly with the objective function in (1), and typically involve various relaxations to optimizations problems that can be efficiently solved. An example of such an approach is the linear programming method

proposed by Almohamad and Duffuaa in [AS93]. They considered the 1-norm as the matching criterion for a permutation matrix $P \in \mathcal{P}$:

$$F_0'(P) = ||A_G - PA_H P^T||_1 = ||A_G P - PA_H||_1. \tag{6}$$

The linear program relaxation is obtained by optimizing $F_0'(P)$ on the set of doubly stochastic matrices $\mathcal{D}$ instead of $\mathcal{P}$:

$$\min_{P \in \mathcal{D}} F_0'(P), \tag{7}$$

where the 1-norm of a matrix is defined as the sum of the absolute values of all the elements of a matrix. A priori the solution of (7) is an arbitrary doubly stochastic matrix $X \in \mathcal{D}$, so the final step is a projection of X on the set of permutation matrices (we let denote $\Pi_{\mathcal{P}} X$ the projection of X onto $\mathcal{P}$) :

$$P^* = \Pi_{\mathcal{P}} X = \arg\min_{P \in \mathcal{P}} ||P - X||_F^2,$$

or equivalently:

$$P^* = \arg\max_{P \in \mathcal{P}} X^T P. \tag{8}$$

The projection (8) can be performed with the Hungarian algorithm, with a complexity cubic in the dimension of the problem. The main disadvantage of this method is that the dimensionality (i.e., number of variables and number of constraints) of the linear program (8) is $O(N^2)$, and therefore it is quite hard to process graphs of size more than one hundred nodes.

Other convex relaxations of (1) can be found in [DTC01] and [SA96]. In the next section we describe our new algorithm which is based on the technique of convex-concave relaxations of the initial problems (1) and (3).

3 Convex-concave relaxation

Let us start the description of our algorithm for unlabeled weighted graphs. The generalization to labeled weighted graphs is presented in Section 3.7. The criterion of graph matching problem we consider for unlabeled graphs is the square of the Frobenius norm of the difference between adjacency matrices (1). Since permutation matrices are also orthogonal matrices (i.e., $PP^T = I$ and $P^T P = I$), we can rewrite $F_0(P)$ on $\mathcal{P}$ as follows:

$$F_0(P) = ||A_G - PA_H P^T||_F^2 = ||(A_G - PA_H P^T)P||_F^2 = ||A_G P - PA_H||_F^2. \tag{9}$$

The graph matching problem is then the problem of minimizing $F_0(P)$ over $\mathcal{P}$, which we call **GM**:

$$\textbf{GM}: \min_{P \in \mathcal{P}} F_0(P). \tag{10}$$

3.1 Convex relaxation

A first relaxation of **GM** is obtained by expanding the convex quadratic function $F_0(P)$ on the set of doubly stochastic matrices $\mathcal{D}$:

$$\textbf{QCV}: \min_{P \in \mathcal{D}} F_0(P). \tag{11}$$

The **QCV** problem is a convex quadratic program that can be solved in polynomial time, e.g., by the Frank-Wolfe algorithm [FW56] (see Section 3.5 for more details). However, the optimal value

is usually not an extreme points of $\mathcal{D}$, and therefore not a permutation matrix. If we want to use only **QCV** for the graph matching problem, we therefore have to project its solution on the set of permutation matrices, and to make, e.g., the following approximation:

$$\arg\min_{\mathcal{P}} \ F_0(P) \approx \Pi_{\mathcal{P}} \arg\min_{\mathcal{D}} \ F_0(P) . \tag{12}$$

Although the projection $\Pi_{\mathcal{P}}$ can be made efficiently in $O(N^3)$ by the Hungarian algorithm, the difficulty with this approach is that if $\arg\min_{\mathcal{D}} \ F_0(P)$ is far from $\mathcal{P}$ then the quality of the approximation (12) may be poor: in other words, the work performed to optimize $F_0(P)$ is partly lost by the projection step which is independent of the cost function. The PATH algorithm that we present later can be thought of as a improved projection step that takes into account the cost function in the projection.

3.2 Concave relaxation

We now present a second relaxation of **GM**, which results in a concave minimization problem. For that purpose, let us introduce the diagonal degree matrix D of an adjacency matrix A, which is the diagonal matrix with entries $D_{ii} = d(i) = \sum_{i=1}^{N} A_{ij}$ for $i = 1, \ldots, N$, as well as the Laplacian matrix $L = D - A$. A having only nonnegative entries, it is well-known that the Laplacian matrix is positive semidefinite [Chu97]. We can now rewrite $F_0(P)$ as follows:

$$
\begin{aligned}
F_0(P) &= ||A_G P - P A_H||_F^2 \\
&= ||(D_G P - P D_H) - (L_G P - P L_H)||_F^2 \\
&= ||D_G P - P D_H||_F^2 - 2\mathrm{tr}[(D_G P - P D_H)^T(L_G P - P L_H)] + ||L_G P - P L_H||_F^2 .
\end{aligned} \tag{13}
$$

Let us now consider more precisely the second term in this last expression:

$$
\begin{aligned}
\mathrm{tr}[(D_G P - P D_H)^T(L_G P - P L_H)] &= \underbrace{\mathrm{tr} P P^T D_G^T L_G}_{-\sum d_G^2(i)} + \underbrace{\mathrm{tr} L_H D_H^T P^T P}_{-\sum d_G^2(i)} - \underbrace{\mathrm{tr} P^T D_G^T P L_H}_{-\sum d_G(i)d_H(i)} - \underbrace{\mathrm{tr} D_H^T P^T L_G P}_{-\sum d_H(i)d_G(i)} \\
&= -\sum(d_G(i) - d_H(i))^2 \\
&= ||D_G - D_H||_F^2 \\
&= ||D_G P - P D_H||_F^2 .
\end{aligned} \tag{14}
$$

Plugging (14) into (13) we obtain

$$
\begin{aligned}
F_0(P) &= ||D_G P - P D_H||_F^2 - 2||D_G P - P D_H||_F^2 + ||L_G P - P L_H||_F^2 \\
&= -||D_G P - P D_H||_F^2 + ||L_G P - P L_H||_F^2 \\
&= -\sum_{i,j} P_{ij}(D_G(i) - D_H(j))^2 + \mathrm{tr}(\underbrace{P P^T}_{I} L_G^T L_G) + \mathrm{tr}(L_H^T \underbrace{P^T P}_{I} L_H) - 2 \underbrace{\mathrm{tr}(P^T L_G^T P L_H)}_{\mathrm{vec}(P)^T (L_H \otimes L_G) \mathrm{vec}(P)} \\
&= -\mathrm{tr}(\Delta P) + \mathrm{tr}(L_G^2) + \mathrm{tr}(L_H^2) - 2\mathrm{vec}(P)^T(L_H \otimes L_G)\mathrm{vec}(P) ,
\end{aligned} \tag{15}
$$

where we introduced the matrix $\Delta_{i,j} = (D_H(j,j) - D_G(i,i))^2$ and we used $\otimes$ to denote the Kronecker product of two matrices[1].

---

[1]Definition of the Kronecker product and some important properties may be found in the appendix B.

Let us denote $F_1(P)$ the part of (15) which depends on P:

$$F_1(P) = -\text{tr}(\Delta P) - 2\text{vec}(P)^T(L_H \otimes L_G)\text{vec}(P). \tag{16}$$

From (15) we see that the permutation matrix which minimizes F_1 over $\mathcal{P}$ is the solution of the graph matching problem. Now, minimizing $F_1(P)$ over $\mathcal{D}$ gives us a relaxation of (10) on the set of doubly stochastic matrices. Since graph Laplacian matrices are positive semi-definite, the matrix $L_H \otimes L_G$ is also positive semidefinite as a Kronecker product of two symmetric positive semi-definite matrices [GL96]. Therefore the function $F_1(P)$ is concave on $\mathcal{D}$, and we obtain a concave relaxation of the graph matching problem:

$$\textbf{QCC}: \min_{P \in \mathcal{D}} F_1(P). \tag{17}$$

Interestingly, the global minimum of a concave function is necessarily located at a boundary of the convex set where it is minimized [Roc70], so the minimium of $F_1(P)$ on $\mathcal{D}$ is in fact in $\mathcal{P}$.

At this point, we have obtained two relaxations of **GM** as optimization problems on $\mathcal{D}$: the first one is the convex minimization problem **QCV** (11), which can be solved efficiently but leads to a solution in $\mathcal{D}$ that must then be projected onto $\mathcal{P}$, and the other is the concave minimization problem **QCC** (17) which does not have an efficient (polynomial) optimization algorithm but has the same solution as the initial problem **GM**. We note that these convex and concave relaxation of the graph matching problem can be more generally derived for any quadratic assignment problems [AB01].

3.3   PATH algorithm

We propose to approximatively solve **QCC** by tracking a path of local minima over $\mathcal{D}$ of a series of functions that linearly interpolate between $F_0(P)$ and $F_1(P)$, namely:

$$F_\lambda(P) = (1 - \lambda)F_0(P) + \lambda F_1(P),$$

for $0 \leq \lambda \leq 1$. For all $\lambda \in [0,1]$, F_λ is a quadratic function (which is in general neither convex nor concave for λ away from zero or one). We recover the convex function F_0 for $\lambda = 0$, and the concave function F_1 for $\lambda = 1$. Our method searches sequentially local minima of F_λ, where λ moves from 0 to 1. More precisely, we start at $\lambda = 0$, and find the unique local minimum of F_0 (which is in this case its unique global minimum) by any classical QP solver. Then, iteratively, we find a local minimum of $F_{\lambda+d\lambda}$ given a local minimum of F_λ by performing a local optimization of $F_{\lambda+d\lambda}$ starting from the local minimum of F_λ, using for example the Frank-Wolfe algorithm. Repeating this iterative process for $d\lambda$ small enough we obtain a path of solutions $P^*(\lambda)$, where $P^*(0) = \arg\min_{P \in \mathcal{D}} F_0(P)$ and $P^*(\lambda)$ is a local minimum of F_λ for all $\lambda \in [0,1]$. Noting that any local minimum of the convex function F_1 on $\mathcal{D}$ is in $\mathcal{P}$, we finally output $P^*(1) \in \mathcal{P}$ as an approximate solution of **GM**.

The pseudo-code for this PATH algorithm is presented in Figure 2. The rationale behind it is that among the local minima of $F_1(P)$ on $\mathcal{D}$, we expect the one connected to the global minimum of F_0 through path of local minima to be a good approximation of the global minima. Such a situation is for example shown in Figure 3, where in 1 dimension the global minimum of a concave quadratic function on an interval (among two candidate points) can be found by following the path of local minima connected to the unique global minimum of a convex function.

More precisely, and although we do not have any formal result about the optimality of the PATH optimization method, we can mention a few interesting properties of this method:

1. **Initialization**:

    (a) $\lambda := 0$

    (b) $P^*(0) = \arg\min F_0$ — convex optimization problem, global minimum is found by Frank-Wolfe algorithm.

2. **Cycle over** λ:
    do

    (a) $\lambda_{new} := \lambda + d\lambda$

    (b) if $|F_{\lambda_{new}}(P^*(\lambda)) - F_{\lambda_{new}}(P^*(\lambda))| < \epsilon$ then
    $$\lambda = \lambda_{new}$$
    else
    $$P^*(\lambda_{new}) = \arg\min F_{\lambda_{new}} \text{ is found by Frank-Wolfe starting from } P^*(\lambda)$$
    $$\lambda = \lambda_{new}$$
    endif

    while $\lambda < 1$

3. **Output**: $P^{out} := P^*(1)$

Figure 2: Schema of the PATH algorithm

- We know from (15) that for $P \in \mathcal{P}$, $F_1(P) = F_0(P) - \kappa$, where $\kappa = \operatorname{tr}(L_G^2) + \operatorname{tr}(L_H^2)$ is a constant independent of P. As a result, it holds for all $\lambda \in [0, 1]$ that, for $P \in \mathcal{P}$:

$$F_\lambda(P) = F_0(P) - \lambda\kappa.$$

  This shows that if for some λ the global minimum of $F_\lambda(P)$ over $\mathcal{D}$ lies in $\mathcal{P}$, then this minimum is also the global minimum of $F_0(P)$ over $\mathcal{P}$ and therefore the optimal solution of the initial problem. Hence, if for example the global minimum of F_λ is found on $\mathcal{P}$ by the PATH algorithm (for instance, if F_λ is still convex), then the PATH algorithm leads to the global optimum of F. This situation can be seen in the toy example in Figure 3 where, for $\lambda = 0.3$, F_λ has its unique minimum at the boundary of the domain.

- The sub-optimality of the PATH algorithm comes from the fact that, when λ increases, the number of local minima of F_λ may increase and the sequence of local minima tracked by PATH may not be global minima. However we can expect the local minima followed by the PATH algorithm to be interesting approximations for the following reason. First observe that if P_1 and P_2 are two local minima of F_λ for some $\lambda \in [0, 1]$, then the restriction of F_λ to (P_1, P_2) being a quadratic function it has to be concave and P_1 and P_2 must be on the boundary of $\mathcal{D}$. Now, let λ_1 be the smallest λ such that F_λ has several local minima on $\mathcal{D}$. If P_1 denotes the local minima followed by the PATH algorithm, and P_2 denotes the "new" local minimum of F_{λ_1}, then necessarily the restriction of F_{λ_1} to (P_1, P_2) must be concave and have a vanishing derivative in P_2 (otherwise, by continuity of F_λ in λ, there would be a local minimum of F_λ near P_2 for λ slightly smaller than λ_1). Consequently we necessarily have $F_{\lambda_1}(P_1) < F_{\lambda_1}(P_2)$. This situation is illustrated in Figure 3 where, when the second local minimum appears for $\lambda = 0.75$, it is worse than the one tracked by the PATH
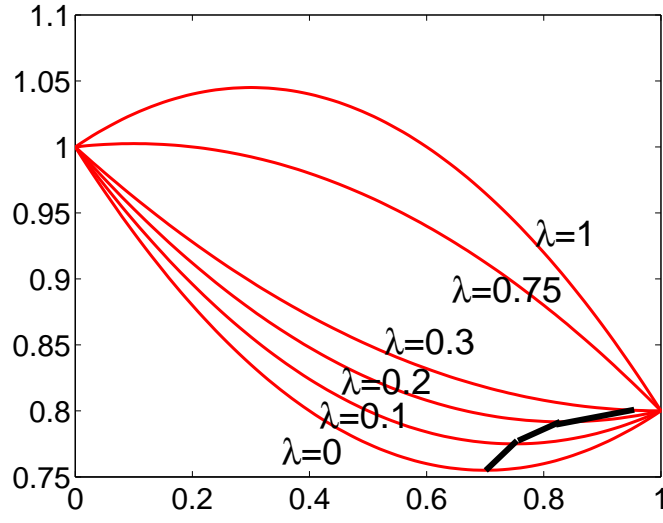
9

Figure 3: Illustration for path optimization approach. F_0 ($\lambda = 0$) — initial convex function, F_1 ($\lambda = 1$) — initial concave function, bold black line — path of function minima $P^*(\lambda)$ ($\lambda = 0 \ldots 0.1 \ldots 0.2 \ldots 0.3 \ldots 0.75 \ldots 1$)

algorithm. More generally, when "new" local minima appears, they are strictly worse than the one tracked by the PATH algorithm. Of course, they may become better than the PATH solution when λ continues to increase.

Of course, in spite of these justifications the PATH algorithm only gives an approximation of the global minimum in the general case. In Appendix A, we give two simple examples when the PATH algorithm respectively succeeds and fails to find the global minimum of the graph matching problem.

3.4   Numerical continuation method interpretation

Our path following algorithm may be considered as a particular case of numerical continuation methods (sometimes called path following methods) [AK90]. These allow to estimate curves given in the following implicit form:

$$T(u) = 0 \text{ where } T \text{ is a mapping } T : R^{K+1} \to R^K . \tag{18}$$

In fact, our PATH algorithm corresponds to a particular implementation of the so-called Generic Predictor Corrector Approach [AK90] widely used in numerical continuation methods.

In our case, we have a set of problems $\min_{P \in \mathcal{D}} (1-\lambda)F_0(P) + \lambda F_1(P)$ parametrized by $\lambda \in [0,1]$. In other words for each λ we have to solve the following system of Karush-Kuhn-Tucker (KKT) equations:

$$
\begin{aligned}
(1-\lambda)\nabla_P F_0(P) + \lambda \nabla_P F_1(P) + B^T \nu + \mu_S &= 0, \\
B\mathrm{P} - 1_{2N} &= 0, \\
P_S &= 0,
\end{aligned}
$$

where S is a set of active constraints, i.e., of pairs of indices (i,j) that satisfy $P_{ij} = 0$, $B\mathrm{P} - 1_{2N} = 0$ codes the conditions $\sum_j P_{ij} = 1\ \forall i$ and $\sum_i P_{ij} = 1\ \forall j$, ν and μ_S are dual variables. We have to

solve this system for all possible sets of active constraints S on the open set of matrices P that satisfy $P_{i,j} > 0$ for $(i,j) \notin S$, in order to define the set of stationary points of the functions F_λ. Now if we let $T(P, \nu, \mu, \lambda)$ denote the left-hand part of the KKT equation system then we have exactly (18) with $K = N^2 + 2N + \#S$. From the implicit function theorem [Mil69], we know that for each set of constraints S,

$$W_S = \{(P, \nu, \mu_S, \lambda) : T(P, \nu, \mu_S, \lambda) = 0 \text{ and } T'(P, \nu, \mu_S, \lambda) \text{ has the maximal possible rank}\} \quad (19)$$

is a smooth 1-dimensional curve or the empty set and can be parametrized by λ. In term of the objective function $F_\lambda(P)$, the condition on $T'(P, \nu, \mu_S, \lambda)$ may be interpreted as a prohibition for the projection of $F_\lambda(P)$ on any feasible direction to be a constant. Therefore the whole set of stationary points of $F_\lambda(P)$ when λ is varying from 0 to 1 may be represented as a union $W(\lambda) = \cup_S W_S(\lambda)$ where each $W_S(\lambda)$ is homotopic to a 1-dimensional segment. The set $W(\lambda)$ may have quite complicated form. Some of $W_S(\lambda)$ may intersect each other, in this case we observe a bifurcation point, some of $W_S(\lambda)$ may connect each other, in this case we have a transformation point of one path into another, some of $W_S(\lambda)$ may appear only for $\lambda > 0$ and/or disappear before λ reaches 1. At the beginning the PATH algorithm starts from $W_\emptyset(0)$, then it follows $W_\emptyset(\lambda)$ until the border of $\mathcal{D}$ (or a bifurcation point). If such an event occurs before $\lambda = 1$ then PATH moves to another segment of solutions corresponding to different constraints S, and keeps moving along segments and sometimes jumping between segments until $\lambda = 1$. As we said in the previous section one of the interesting properties of PATH algorithm is the fact that if $W_S^*(\lambda)$ appears only when $\lambda = \lambda_1$ and $W_S^*(\lambda_1)$ is a local minimum then the value of the objective function F_{λ_1} in $W_S^*(\lambda_1)$ is greater than in the point traced by the PATH algorithm.

3.5   Some implementation details

In this section we provide a few details relevant for the efficient implementation of the PATH algorithms.

Frank-Wolfe   Among the different optimization techniques for the optimization of $F_\lambda(P)$ starting from the current local minimum tracked by the PATH algorithm, we use in our experiments the Frank-Wolfe algorithm which is particularly suited to optimization over doubly stochastic matrices [D.B99]. The idea of the this algorithm is to sequentially minimize linear approximations of $F_0(P)$. Each step includes three operations:

1. estimation of the gradient $\nabla F_\lambda(P_n)$,

2. resolution of the linear program $P_n^* = \arg\min_{P \in D} \langle \nabla F_\lambda(P_n), P \rangle$,

3. line search: finding the minimum of $F_\lambda(P)$ on the segment $[P_n \ P_n^*]$.

An important property of this method is that the second operation can be done efficiently by the Hungarian algorithm, in $O(N^3)$.

Efficient gradient computations   Another essential point is that we do not need to store matrices of size $N^2 \times N^2$ for the computation of $\nabla F_\lambda(P)$, because the product in $\nabla F_\lambda(P) = -\text{vec}(\Delta^T) - 2(L_H \otimes L_G)\text{vec}(P)$ can be expressed in terms of $N \times N$ matrices and Kronecker products:

$$\nabla F_1(P) = -\text{vec}(\Delta^T) - 2(L_H \otimes L_G)\text{vec}(P) = -\text{vec}(\Delta^T) - 2\text{vec}((L_G P L_H^T)).$$

Initialization The proposed algorithm can be accelerated by the application of Newton algorithm as the first step of **QCV** (minimization of $F_0(P)$). First, let us rewrite the **QCV** problem as follows:

$$\min_{P \in \mathcal{D}} \|A_G P - P A_H\|_F^2 \Leftrightarrow$$

$$\min_{P \in \mathcal{D}} \|(I \otimes A_G - A_H^T \otimes I)\text{vec}(P)\|_F^2 \Leftrightarrow$$

$$\min_{P \in \mathcal{D}} \text{vec}(P)^T \underbrace{(I \otimes A_G - A_H^T \otimes I)^T (I \otimes A_G - A_H^T \otimes I)}_{Q} \text{vec}(P) \Leftrightarrow$$

$$\min_{P \in \mathcal{D}} \text{vec}(P)^T Q \text{vec}(P) \Leftrightarrow \begin{cases} \min_P \text{vec}(P)^T Q \text{vec}(P) \text{ such that} \\ \qquad B\text{vec}(P) = 1_{2N} \\ \qquad \text{vec}(P) \geq 0_{N^2} \end{cases} \tag{20}$$

where B is the matrix which codes the conditions $\sum_j P_{i,j} = 1$ and $\sum_i P_{i,j} = 1$. The Lagrangian has the following form

$$\mathcal{L}(P, \nu, \lambda) = \text{vec}(P)^T Q \text{vec}(P) + \nu^T (B\text{vec}(P) - 1_{2N}) + \mu^T \text{vec}(P), \tag{21}$$

where ν and μ are Lagrange multipliers. Now we would like to use Newton method for constrained optimization [D.B99] to solve (20). Let μ_a denote the set of variables associated to the set of active constraints $\text{vec}(P) = 0$ at the current points, then the Newton step consist in solving the following system of equations:

$$\begin{bmatrix} 2Q & B^T & I_a \\ B & 0 & 0 \\ I_a & 0 & 0 \end{bmatrix} \begin{bmatrix} \text{vec}(P) \\ \nu \\ \mu_a \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \quad \begin{array}{l} N^2 \text{ elements,} \\ 2N \text{ elements,} \\ \text{number of active inequality constraints.} \end{array} \tag{22}$$

More precisely we have to solve (22) for P. The problem is that in general situations this problem is computationally demanding because it involves the inversion of matrices of size $O(N^2) \times O(N^2)$. In some particular cases, however, the Newton step becomes feasible. Typically, if none of the constraints $\text{vec}P \geq 0$ are active, then (22) takes the following form[2]:

$$\begin{bmatrix} 2Q & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} \text{vec}(P) \\ \nu \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad \begin{array}{l} N^2 \text{ elements}, \\ 2N \text{ elements}. \end{array} \tag{23}$$

The solution is then obtained as follows:

$$\text{vec}(P)_{KKT} = \frac{1}{2} Q^{-1} B^T (B Q^{-1} B^T)^{-1} 1_{2N}. \tag{24}$$

Because of the particular form of matrices Q and B, the expression (24) may be computed very simply with the help of Kronecker product properties in $O(N^3)$ instead of $O(N^6)$. More precisely, the first step is the calculation of $M = B Q^{-1} B^T$ where $Q = (I \otimes B_G - B_H \otimes I)^2$. The matrix Q^{-1} may be represented as follows:

$$Q^{-1} = (U_H \otimes U_G)(I \otimes \Lambda_G - \Lambda_H \otimes I)^{-2}(U_H \otimes U_G)^T. \tag{25}$$

---

[2]It is true if we start our algorithm, for example, from the constant matrix $P_0 = \frac{1}{N} 1_N 1_N^T$.

Therefore the (i, j)-th element of M is the following product:

$$B_i Q^{-1} B_j^T = \text{vec}(U_H^T \widetilde{B_i}^T U_G)^T)(\Lambda_G - \Lambda_H)^{-2}\text{vec}(U_G^T \widetilde{B_j}^T U_H), \qquad (26)$$

where B_i is the i-th row of B and $\widetilde{B_i}$ is B_i reshaped into a $N \times N$ matrix. The second step is an inversion of the $2N \times 2N$ matrix M, and a sum over columns $M^s = M^{-1}1_{2N}$. The last step is a multiplication of Q^{-1} by $B^T M^s$, which can be done with the same tricks as the first step. The result is the value of matrix P_{KKT}. We then have two possible scenarios:

1. If $P_{KKT} \in \mathcal{D}$, then we have found the solution of (20).

2. Otherwise we take the point of intersection of the line (P_0, P_{KKT}) and the border $\partial\mathcal{D}$ as the next point and we continue with Frank-Wolfe algorithm. Unfortunately we can do the Newton step only once, then some of $P \geq 0$ constraints become active and efficient calculations are not feasible anymore. But even in this case the Newton step is generally very useful because it decreases a lot the value of the objective function.

3.6 Algorithm complexity

Here we present the complexity of the algorithms discussed in the paper.

- Umeyama's algorithm has three components: matrix multiplication, calculation of eigenvectors and application of the Hungarian algorithm for (5). Complexity of each component is equal to $O(N^3)$. Thus Umeyama's algorithm has complexity $O(N^3)$.

- LP approach (7) has complexity $O(N^7)$ (worst case) because it may be rewritten as an linear optimization problem with $3N^2$ variables [BV03].

- Each step of the path algorithm has complexity $O(N^3)$: multiplication of $N \times N$ matrices, eigendecompostion problem and Hungarian algorithm. In addition the complexity of path algorithm depends on the number of iterations, this number is a function of the stopping criterion (for instance, value of the gradient).

3.7 Vertex pairwise similarities

If we match two labeled graphs, then we may increase the performance of our method by using information on pairwise similarities between their nodes. In fact one method of image matching uses only this type of information, namely shape context matching [BMP02]. To integrate the information on vertex similarities we use the approach proposed in (3), but in our case we use $F_\lambda(P)$ instead of $F_0(P)$

$$\min_P \ F_\lambda^\alpha(P) = \min_P \ (1 - \alpha)F_\lambda(P) + \alpha\text{tr}(C^T P), . \qquad (27)$$

The advantage of the last formulation is that $F_\lambda^\alpha(P)$ is just $F_\lambda(P)$ with an additional linear term. Therefore we can use the same algorithm for the minimization of $F_\lambda^\alpha(P)$ as the one we presented for the minimization of $F_\lambda(P)$.

3.8 Matching graphs of different sizes

Often in practice we have to match graphs of different sizes N_G and N_H (suppose, for example, that $N_G > N_H$). In this case we have to match all vertices of graph H to a subset of vertices of graph G. In the usual case when $N_G = N_H$ the error (1) corresponds to the number of mismatched edges (edges which exist in one graph and do not exist in the other one). When we match graphs of different sizes the situation is a bit more complicated. Let $V_G^+ \subset V_G$ denote the set of vertices of graph G that are selected for matching to vertices of graph H, let $V_G^- = V_G \setminus V_G^+$ denote all the rest. Therefore all edges of the graph G are divided into four parts $E_G = E_G^{++} \cup E_G^{+-} \cup E_G^{-+} \cup E_G^{--}$, where E_G^{++} are edges between vertices from V_G^+, E_G^{--} are edges between vertices from V_G^-, E_G^{+-} and E_G^{-+} are edges from V_G^+ to V_G^- and from V_G^- to V_G^+ respectively. For undirected graphs the sets E_G^{+-} and E_G^{-+} are the same (but, for directed graphs we do not consider, they would be different). The edges from E_G^{--}, E_G^{+-} and E_G^{-+} are always mismatched and a question is whether we have to take them into account in the objective function or not. According to the answer we have three types of matching error (four for directed graphs) with interesting interpretation.

1. We count only the number of mismatched edges between H and the chosen subgraph $G^+ \subset G$. It corresponds to the case when the matrix P from (1) is a matrix of size $N_G \times N_H$ and $N_G - N_H$ rows of P contain only zeros.

2. We count the number of mismatched edges between H and chosen subgraph $G^+ \subset G$. And we also count all edges from E_G^{--}, E_G^{+-} and E_G^{-+}. In this case P from (1) is a matrix of size $N_G \times N_G$. And we transform matrix A_H into a matrix of size of size $N_G \times N_G$ by adding $N_G - N_H$ zero rows and zero columns. It means that we add dummy isolated vertices to the smallest graph, and then we match graphs of the same size.

3. We count the number of mismatched edges between H and chosen subgraph $G^+ \subset G$. And we also count all edges from E_G^{+-} (or E_G^{-+}). It means that we count matching error between H and G^+ and we count also the number of edges which connect G^+ and G^-. In other words we are looking for subgraph G^+ which is similar to H and which is maximally isolated in the graph G.

Each type of error may be useful according to context and interpretation, but a priori, it seems that the best choice is the second one where we add dummy nodes to the smallest graph. The main reason is the following. Suppose that graph H is quite sparse, and suppose that graph G has two candidate subgraphs G_s^+ (also quite sparse) and G_d^+ (dense). The upper bound for the matching error between H and G_s^+ is $\#V_H + \#V_{G_s^+}$, the lower bound for the matching error between H and G_d^+ is $\#V_{G_d^+} - \#V_H$. So if $\#V_H + \#V_{G_s^+} < \#V_{G_d^+} - \#V_H$ then we will always choose the graph G_s^+ with the first strategy, even if it is not similar at all to the graph H. The main explanation of this effect lies in the fact that the algorithm tries to minimize the number of mismatched edges, and not to maximize the number of well matched edges. In contrast, when we use dummy nodes, we do not have this problem because if we take a very sparse subgraph G^+ it increases the number of edges in G^- (the common number of edges in G^+ and G^- is constant and is equal to the number of edges in G) and finally it decreases the quality of matching.

4 Simulations

4.1 Synthetic examples

In this section we compare the proposed algorithm with some classical methods on artificially generated graphs. Our choice of random graph types is based on [NSW01] where authors discuss different types of random graphs which are the most frequently observed in various real world applications (world wide web, collaborations networks, social networks, etc...). Each type of random graphs is defined by the distribution function of node degree Prob(node degree $= k) = VD(k)$. The vector of node degrees of each graph is supposed to be an i.i.d sample from $VD(k)$. In our experiments we have used the following types of random graphs:

Binomial law	Geometric law	Power law
$VD(k) = C_N^k p^k (1-p)^{1-k}$	$VD(k) = (1 - e^{-\mu})e^{\mu k}$	$VD(k) = C_\tau k^{-\tau}$

The schema of graph generation is the following

1. generate a sample $d = (d_1, \ldots, d_N)$ from $VD(k)$

2. if $\sum_i d_i$ is odd then goto step 1

3. while $\sum_i d_i > 0$

   (a) choose randomly two non-zero elements from d: d_{n1} and d_{n2}

   (b) add edge $(n1, n2)$ to the graph

   (c) $d_{n1} \leftarrow d_{n1} - 1i \quad d_{n2} \leftarrow d_{n2} - 1$

If we are interested in isomorphic graph matching then we compare just the initial graph and its randomly permuted copy. To test the matching of non-isomorphic graphs, we add randomly $\sigma_n N_E$ edges to the initial graph and to its permitted copy, where N_E is the number of edges in the original graph, and σ_n is the noise level.

4.2 Results

The first series of experiments are experiments on small size graphs (N=8), here we are interested in comparison of the PATH algorithm (see figure 2), the **QCV** approach (11), Umeyama spectral algorithm (5), the linear programming approach (7) and exhaustive search which is feasible for the small size graphs. The algorithms were tested on the three types of random graphs (binomial, exponential and power). The results are presented in Figure 4. The same experiment was repeated for middle-sized graphs ($N = 20$, Figure 5) and for large graphs ($N = 100$, Figure 6).

In all cases, the PATH algorithm works much better than all other approximate algorithms. There are some important things to note here. First, the choice of norm in (1) is not very important — results of QCV and LP are about the same. Second, following the solution paths is very useful compared to just minimizing the convex relaxation and projecting the solution on the set of permutation matrices (PATH algorithms works much better than QCV). Another noteworthy observation is that the performance of PATH is very close to the optimal solution when the later can be evaluated.

We note that sometimes the matching error decreases as the noise level increases (e.g., in Figures 6c,5c). The explanation is the following. The matching error is upper bounded by the
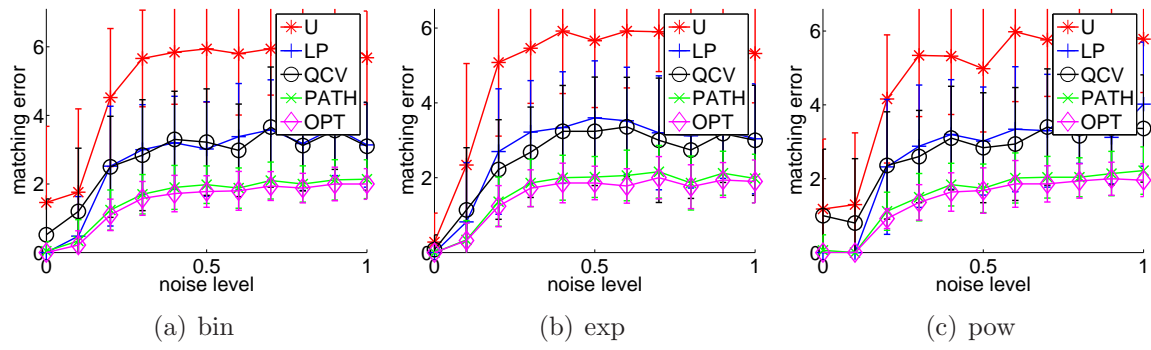
Figure 4: Matching error (mean value over sample of size 100) as a function of noise. Graph size N=8. U — Umeyama's algorithm, LP — linear programming algorithm, QCV — convex optimization, PATH — path minimization algorithm,OPT — an exhaustive search (the global minimum). The range of error bars is the standard deviation of matching errors
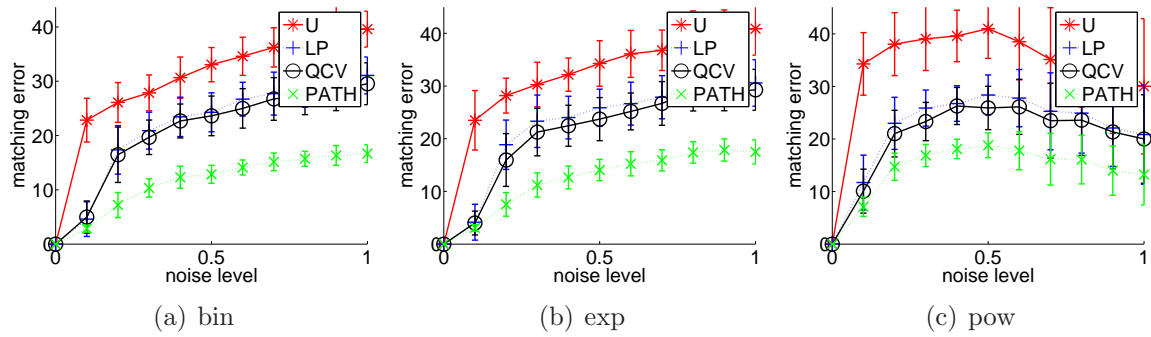

Figure 5: Matching error (mean value over sample of size 100) as a noise function. Graph size N=20. U — Umeyama's algorithm, LP — linear programming algorithm, QCV — convex optimization, PATH — path minimization algorithm.
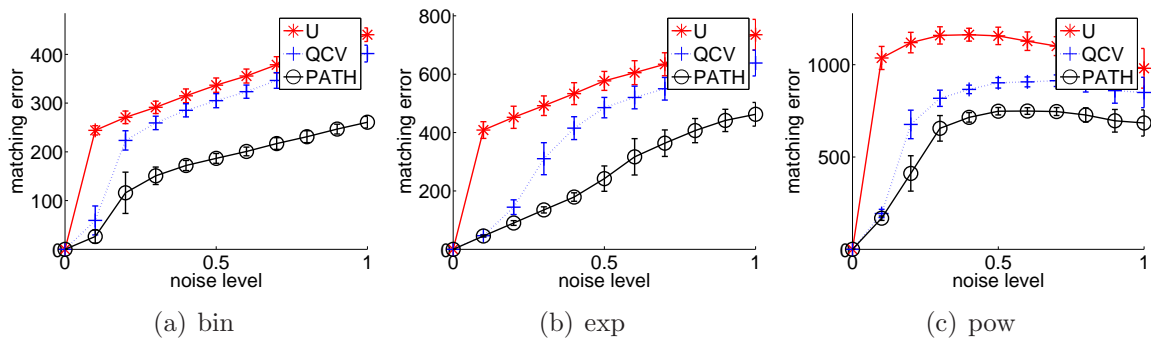

Figure 6: Matching error (mean value over sample of size 100) as a noise function. Graph size N=100. U — Umeyama's algorithm, QCV — convex optimization, PATH — path minimization algorithm.

minimum of the total number of zeros in the adjacency matrices A_G and A_H. So when graphs are dense and noise level increases, it makes graphs even more dense. And therefore the upper bound of matching error decreases.

Another important aspect to compare the different algorithms is their run-time complexity. Figure 7 shows the time needed to obtain the matching between two graphs as a function of the number of vertices N, for the different methods. These curves are coherent with theoretical values of algorithm complexities summarized in Section 3.6. In particular we observe that Umeyama's algorithm is the fastest method, but that QCV and PATH have the same complexity in N. The LP method is competitive with QCV and PATH for small graphs, but has a worse complexity in N.
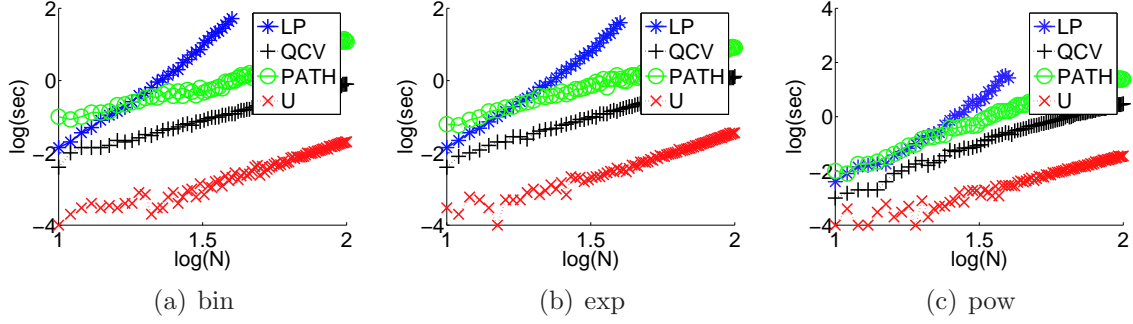
(a) bin   (b) exp   (c) pow

Figure 7: Timing of U,LP,QCV and PATH algorithms as a function of graph size, for the different random graph models. LP slope ≈ 6.7, U, QCV and PATH slope ≈ 3.4

5 QAP benchmark library

The problem of graph matching may be considered as a particular case of the quadratic assignment problem (QAP). The minimization of the loss function (1) is equivalent to the minimization of the following function:

$$\min_{P} \ \mathrm{tr}(P^T A_G^T P A_H). \tag{28}$$

Therefore it is interesting to compare our method with other approximate methods proposed for QAP. [DTC01] proposed the QPB algorithm for that purpose and tested it on matrices from the QAP benchmark library [Cel07], QPB results were compared to the results of graduated assignment algorithm GRAD [SA96] and Umeyama's algorithm. Results of PATH application to the same matrices are presented in Table 1, scores for QPB and graduated assignment algorithm are taken directly from the publication [DTC01]. We observe that on 14 out of 16 benchmark, PATH is the best optimization method among the methods tested.

Table 1: Experiment results for QAPLIB benchmark datasets.

QAP	MIN	PATH	QPB	GRAD	U
chr12c	11156	**18048**	20306	19014	40370
chr15a	9896	**19086**	26132	30370	60986
chr15c	9504	**16206**	29862	23686	76318
chr20b	2298	**5560**	6674	6290	10022
chr22b	6194	**8500**	9942	9658	13118
esc16b	292	300	**296**	298	306
rou12	235528	**256320**	278834	273438	295752
rou15	354210	391270	**381016**	457908	480352
rou20	725522	**778284**	804676	840120	905246
tai10a	135028	**152534**	165364	168096	189852
tai15a	388214	**419224**	455778	451164	483596
tai17a	491812	**530978**	550852	589814	620964
tai20a	703482	**753712**	799790	871480	915144
tai30a	1818146	**1903872**	1996442	2077958	2213846
tai35a	2422002	**2555110**	2720986	2803456	2925390
tai40a	3139370	**3281830**	3529402	3668044	3727478

6 Image processing

In this section, we present two applications in image processing. The first one (Section 6.1) illustrates how taking into account information on graph structure may increase image alignment quality. The second one (Section 6.2) shows that the structure of contour graphs may be very important in classification tasks. In both examples we compare the performance of our method with the shape context approach [BMP02], a state-of-the-art method for image matching.

6.1 Alignment of vessel images

The first example is dedicated to the problem of image alignment. We consider two photos of vessels in human eyes. The original photos and the images of extracted vessel contours (obtained from the method of [WKME03]) are presented in Figure 8. To align the vessel images, the shape context algorithm uses the context radial histograms of contour points (see [BMP02]). In other words, according to the shape context algorithm one aligns points which have similar context histograms. The PATH algorithm uses also information about the graph structure. When we use the PATH algorithm we have to tune the parameter α (27), we tested several possible values and we took the one which produced the best result. To construct graph we use all points of vessel contours as graph nodes and we connect all nodes within a circle of radius r^3. Finally, to each edge (i, j) we associate the weight $w_{i,j} = \exp(-|x_i - y_j|)$.
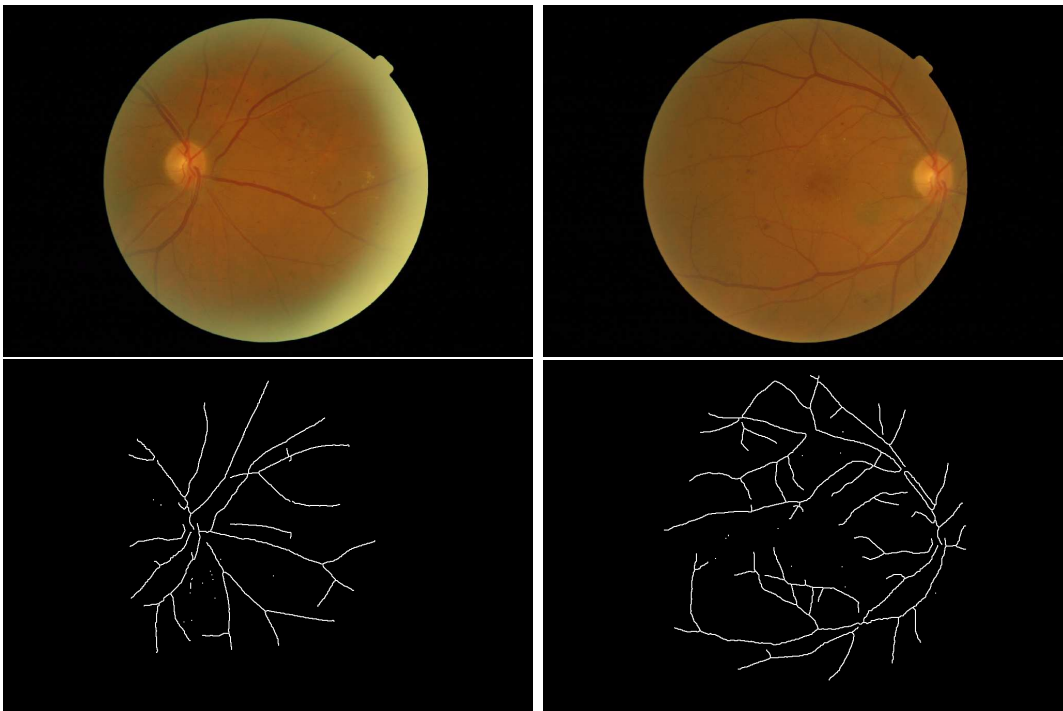

Figure 8: Eye photos (top) and vessel contour extraction (bottom).

A graph matching algorithm produces an alignment of image contours, then to align two images we have to expand this alignment to the rest of image. For this purpose, we use a smooth spline-based transformation [Boo89]. In other words, we estimate parameters of the spline transformation

---

[3]in our case we use $r = 50$.

from the known alignment of contour points and then we apply this transformation to the whole image. Results of image matching based on shape context algorithm and on PATH algorithm are presented in Figure 9, where black lines designate connections between associated points. We observe that the context shape method creates many unwanted matching, while PATH produces a matching that visually corresponds to a correct alignment of the structure of vessels. The main
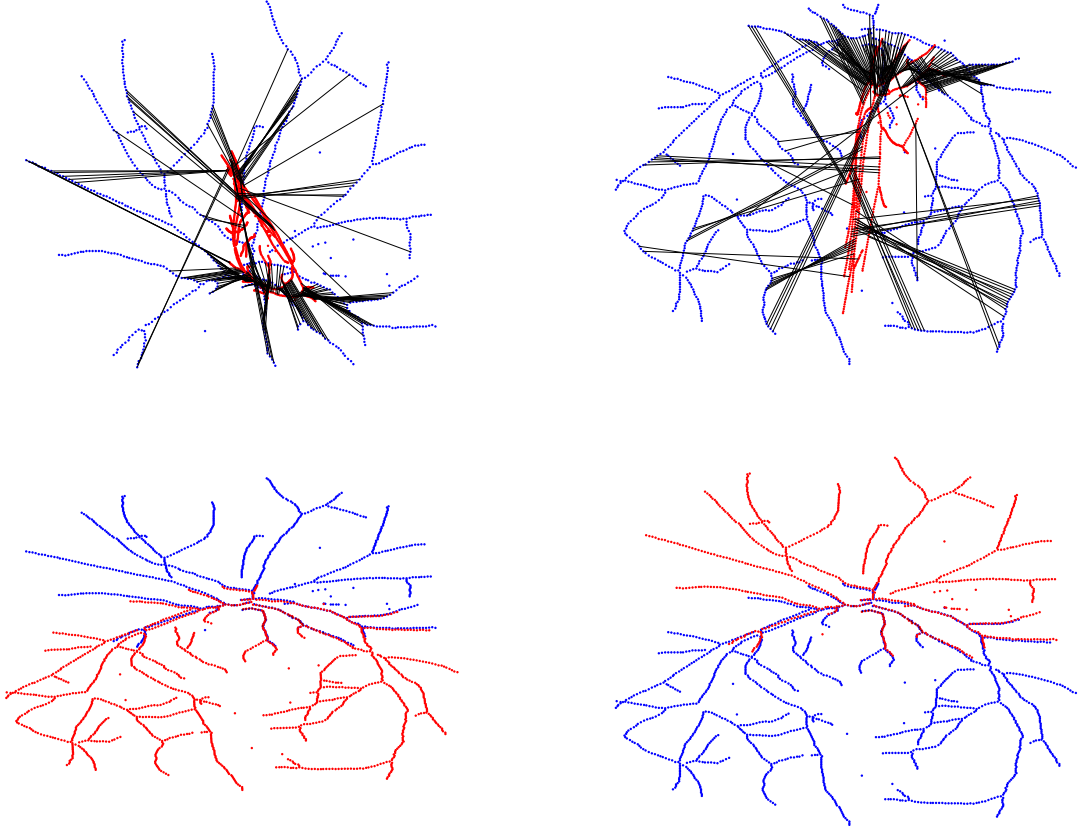

Figure 9: Comparison of alignment based on shape context (top) and alignment based on the PATH optimization algorithm (bottom). For each algorithm we present two alignments: image '1' on image '2' and the inverse. Each alignment is a spline-based transformation (see text).

reason why graph matching works better than shape context matching is the fact that shape context does not take into account the relational positions of matched points and may lead to totally incoherent graph structures. In contrast, graph matching tries to match pairs of nearest points in one image to pairs of nearest points in another one.
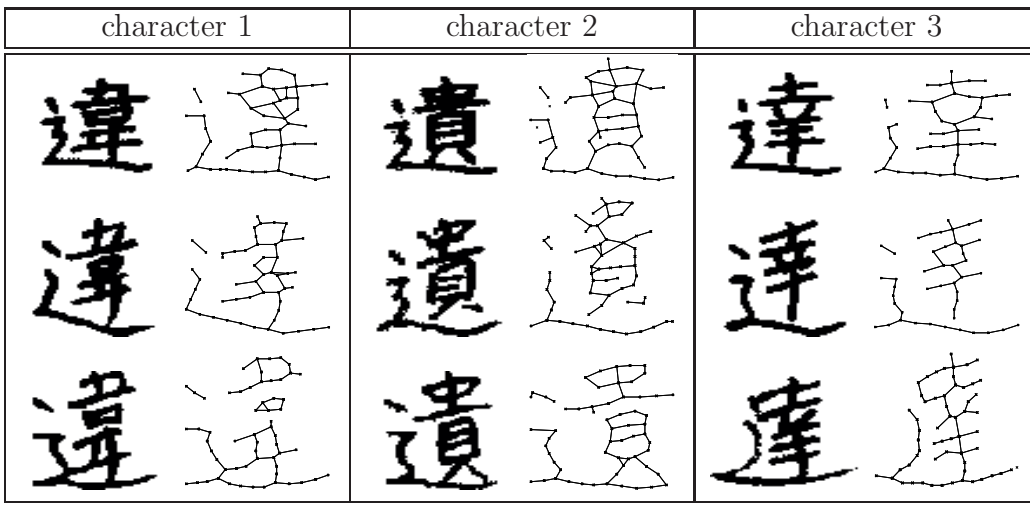
Figure 10: Chinese characters from the ETL9B dataset.

6.2 Recognition of handwritten chinese characters

Another example that we consider in this paper is the problem of chinese character recognition from the ETL9B dataset [SYY85]. The main idea is to use a score of optimal matching as a similarity measure between two images of characters. This similarity measure can be used then in machine learning algorithms, K-nearest neighbors (KNN) for instance, for character classification. Here we compare the performance of four methods: linear support vector machine (SVM), SVM with gaussian kernel, KNN based on score of shape context matching and KNN based on scores from graph matching which combines structural and shape context information. As a score, we use just the value of the objective function (27) at the (locally) optimal point. We have selected three chinese characters known to be difficult to distinguish by automatic methods. Examples of these characters as well as examples of extracted graphs (obtained by thinning and uniformly subsampling the images) are presented in Figure 10. For SVM based algorithms, we use directly the values of image pixels (so each image is represented by a binary vector), in graph matching algorithm we use binary adjacency matrices of extracted graphs and shape context matrices (see [BMP02]).

Our data set consist of 50 exemples (images) of each class. Each image is represented by 63×64 binary matrix. To compare different methods we use the cross validation error (five folds). The dependency of classification error from two algorithm parameters (α — coefficient of linear combination (27) and k — number of nearest neighbors used in KNN)) is shown in Figure 11.

Two extreme choices $\alpha = 1$ and $\alpha = 0$ correspond respectively to pure shape context matching, i.e., when only node labels information is used, and pure unlabeled graph matching. It is worth observing here that KNN based just on the score of unlabeled graph matching does not work very well, the classification error being about 60%. An explanation of this phenomenon is the fact that learning patterns have very unstable graph structure within one class. The pure shape context method has a classification error of about 39%. The combination of shape context and graph structure informations allows to decrease the classification error down to 25%. Complete results can be found in Table 2.
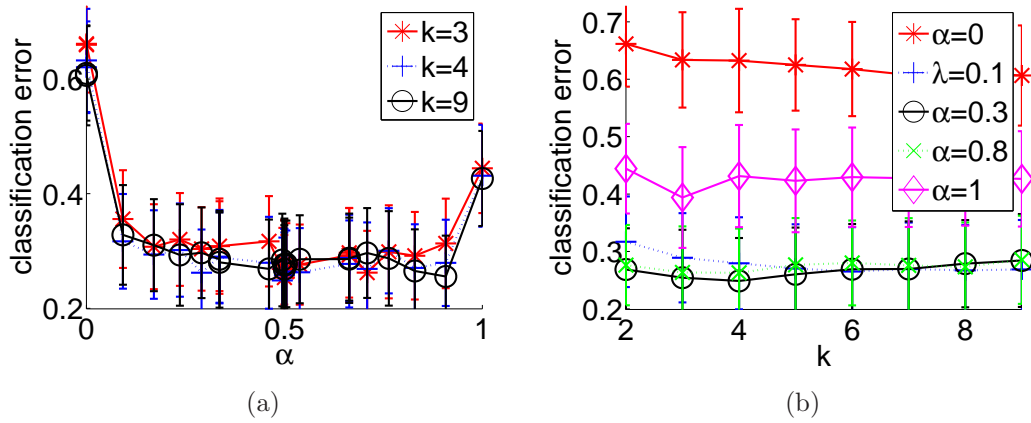
Figure 11: (a) Classification error as a function of α. (b) Classification error as a function of k. Classification error is estimated as cross validation error (five folds, 50 repetition), the range of the error bars is the standard deviation of test error over one fold (not averaged over folds and repetition)

Table 2: Classification of chinese characters. (CV, STD)—mean and standard deviation of test error over cross-validation runs (five folds, 50 repetitions)

Method	CV	STD
Linear SVM	0.377	$\pm$ 0.090
SVM with gaussian kernel	0.359	$\pm$ 0.076
KNN (λ=1): shape context	0.399	$\pm$ 0.081
KNN (λ=0.4)	0.248	$\pm$ 0.075
KNN (λ=0): pure graph matching	0.607	$\pm$ 0.072

22

7 Conclusion

We have presented the PATH algorithm, a new technique for graph matching based on convex-concave relaxations of the initial integer programming problem. PATH allows to integrate the alignment of graph structural elements with the matching of vertices with similar labels. Its results are competitive with state-of-the-art methods in several graph matching and QAP benchmark experiments. Moreover, PATH has a theoretical and empirical complexity competitive with the fastest available graph matching algorithms.

Two points can be mentioned as interesting directions for further research. First, the quality of the convex-concave approximation is defined by the choice of convex and concave relaxation functions. Better performances may be achieved by more appropriate choices of these functions. Second, another interesting point concerns the construction of a good concave relaxation for the problem of directed graph matching, i.e., for asymmetric adjacency matrix. Such generalizations would be interesting also as possible polynomial-time approximate solutions for the general QAP problem.

A A toy example

The goal of this subsection is to emphasize that the PATH algorithm gives only an approximate solution in general, and that there are situations when it fails to locate the global minimum. More precisely, we provide simple examples to highlight the fact that the set of local optima tracked by PATH may not lead to the global minimum of the concave function, because the the global optimum may "jump" from one segment of local minima to another when λ increases.

More precisely, we consider two simple graphs with the following adjacency matrices:

$$\mathbf{G} = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} \text{ and } \mathbf{H} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

Let C denote the cost matrix of vertex association

$$\mathbf{C} = \begin{bmatrix} 0.1691 & 0.0364 & 1.0509 \\ 0.6288 & 0.5879 & 0.8231 \\ 0.8826 & 0.5483 & 0.6100 \end{bmatrix}.$$

Let us suppose that we have fixed the tradeoff $\alpha = 0.5$, and that our objective is then to find the global minimum of the following function:

$$F_0(P) = 0.5\|GP - PH\|_F^2 + 0.5\text{tr}(C'P), \qquad P \in \mathcal{P}. \tag{29}$$

As we said before, the main idea underlying the PATH algorithm is to try to follow the path of global minima of $F_\lambda^\alpha(P)$ (27). It may be possible, if all global minima P_λ^* form a continuous path. But that is not always true. In the case of small graphs we are able to find the exact global minimum of $F_\lambda^\alpha(P)$ for all λ. The trace of global minima as functions of λ is presented in Figure Aa (i.e., we plot the values of the nine parameters of the doubly stochastic matrix, which are, as expected, all equal to zero or one when $\lambda = 1$). When λ is near 0.2 there is a jump of global minimum from one face to another. However if we change the linear term C to

$$\mathbf{C'} = \begin{bmatrix} 0.4376 & 0.3827 & 0.1798 \\ 0.3979 & 0.3520 & 0.2500 \\ 0.1645 & 0.2653 & 0.5702 \end{bmatrix},$$

23

then the trace becomes smooth (see Figure Ab) and the PATH algorithm then finds the globally optimum point. Characterizing cases where the path is indeed smooth is the subject of ongoing research.
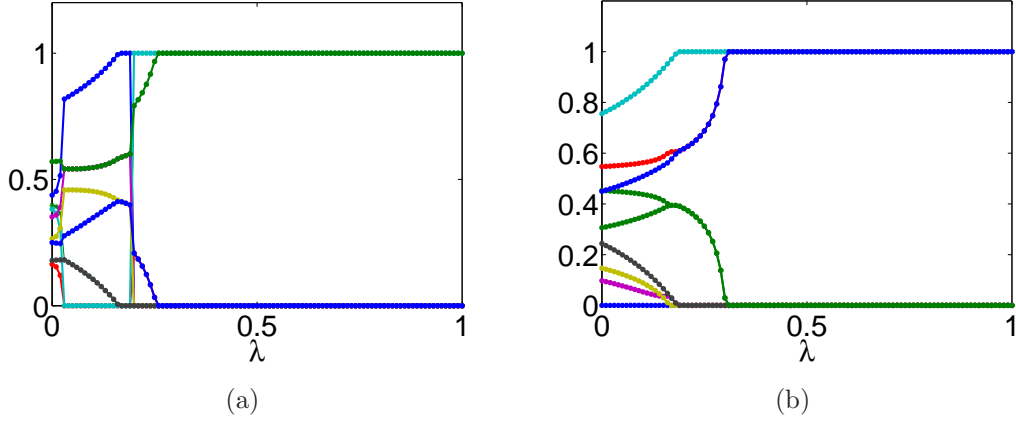

(a)   (b)

Figure 12: Nine coordinates of global minimum of F_λ^α as a function of λ

B   Kronecker product

The Kronecker product of two matrices $A \otimes$ is defined as follows:

$$A \otimes B = \begin{bmatrix} Ba_{11} & \cdots & Ba_{1n} \\ \vdots & \ddots & \vdots \\ Ba_{m1} & \cdots & Ba_{mn} \end{bmatrix}. \tag{30}$$

Two important properties of Kronecker product that we use in this paper are:

$$(A^T \otimes B)\mathrm{vec}(X) = \mathrm{vec}(BXA), \text{ where } \mathrm{vec}(X) = \begin{bmatrix} x_{11} \\ \vdots \\ x_{m1} \\ \vdots \\ x_{mn} \end{bmatrix}, \tag{31}$$

and:

$$\mathrm{tr}(X^T A^T X B) = \mathrm{vec}(X)^T (B \otimes A)\mathrm{vec}(X). \tag{32}$$

References

[AAI95]   A.Filatov, A.Gitis, and I.Kil. Graph-based handwritten digit string recognition. *Third International Conference on Document Analysis and Recognition (ICDAR'95)*, pages 845–848, 1995.

[AB01]   K. M. Anstreicher and N. W. Brixius. A new bound for the quadratic assignment problem based on convex quadratic programming. *Mathematical Programming*, 89(3):341–357, 2001.

[AK90]   E.L. Allgower and K.Georg. *Numerical continuation methods*. Springer, 1990.

[AS93]   H.A. Almohamad and S.O.Duffuaa. A linear programming approach for the weighted graph matching problem. *Transaction on pattern analysis and machine intelligence*, 15, 1993.

[BL00]   J. M. Borwein and A. S. Lewis. *Convex Analysis and Nonlinear Optimization*. Springer-Verlag, New York, 2000.

[BMP02]   Serge Belongie, Jitendra Malik, and Jan Puzicha. Shape matching and object recognition using shape contexts. *Transaction on pattern analysis and machine intelligence*, 24, 2002.

[Boo89]   F.L. Bookstein. Principal warps: thin-plate splines and the decomposition of deformations. *Transaction on pattern analysis and machine intelligence*, 11:567–585, 1989.

[BR00]   B.Luo and Edwin R.Hancock. Alignment and correspondence using sigular value decomposition. *Lecture notes in computer science*, 1876:226–235, 2000.

[BV03]   S. Boyd and L. Vandenberghe. *Convex Optimization*. Camb. Univ. Press, 2003.

[CC05]   C.Schellewald and C.Schnor. Probabilistic subgraph matching based on convex relaxation. *Lecture notes in computer science*, pages 171–186, 2005.

[Cel07]   Eranda Cela. Qaudratuc assignment problem library. *www.opt.math.tu-graz.ac.at/qaplib/*, 2007.

[CFSV91]   L. P. Cordella, P. Foggia, C. Sansone, and M. Vento. Performance evaluation of the vf graph matching algorithm. *Proc. of the 10th ICIAP*, 2:1038–1041, 1991.

[Chu97]   Fan R. K. Chung. *Spectral Graph Theory*. Americal Mathematical Society, 1997.

[CK04]   Terry Caelli and Serhiy Kosinov. An eigenspace projection clustering method for inexact graph matching. *Transaction on pattern analysis and machine intelligence*, 24, 2004.

[CR02]   Marco Carcassoni and Edwin R.Hancock. Spectral correspondance for point pattern matching. *Pattern Recognition*, 36:193–204, 2002.

[D.B99]   D.Bertsekas. *Nonlinear programming*. Athena Scientific, 1999.

[DPCM04]   D.Conte, P.Foggia, C.Sansone, and M.Vento. Thirty years of graph matching in pattern recognition. *International journal of pattern recognition and artificial intelligence*, 18:265–298, 2004.

[DTC01]   D.Cremers, T.Kohlberger, and C.Schnor. Evaluation of convex optimization techniques for the weighted graph-matching problem in computer vision. *Patter Recognition*, 2191, 2001.

[FW56]   M. Frank and P. Wolfe. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3:95–110, 1956.

[GL96]   Gene H. Golub and Charles F. Van Loan. *Matrix computations (3rd ed.).* Johns Hopkins University Press, Baltimore, MD, USA, 1996.

[GM79]   Johnson D.S. Garey M.R. *Computer and intractability: A guide to the theory of NP-completeness.* San Francisco, CA: W. H. Freeman, 1979.

[KME05]   K.Brein, M.Remm, and E.Sonnhammer. Inparanoid: a comprehensive database of eukaryothic orthologs. *Nucleic acids research*, 33, 2005.

[Kuh55]   H.W. Kuhn. The hungarian method for the assignment problem. *Naval Research*, 2:83–97, 1955.

[LL99]   Raymond S . T. Lee and James N. K. Liu. An oscillatory elastic graph matching model for recognition of offline handwritten chinese characters. *Third International Conference on Knowledge-Based Intelligent Information Engineeing Systems*, pages 284–287, 1999.

[McG83]   Leon F. McGinnis. Implementation and testing of a primal-dual algorithm for the assignment problem. *Operations Research*, 31:277–291, 1983.

[Mil69]   J.W. Milnor. *Topology from the Differentiable Viewpoint.* Univ. Press of Virginia, 1969.

[NSW01]   M. E. J. Newman, S. H. Strogatz, and D. J. Watts. Random graphs with arbitrary degree distributions and their applications. *PHYSICAL REVIEW*, 64, 2001.

[RJB07]   R.Singh, J.Xu, and B.Berger. Pairwise global alignment of protein interaction networks by matching neighborhood topology. *Fill in*, 0, 2007.

[Roc70]   R. Rockafeller. *Convex Analysis.* Princeton Univ. Press, 1970.

[SA96]   S.Gold and A.Rangarajan. A graduated assignment algorithm for graph matching. *Transaction on pattern analysis and machine intelligence*, 18, 1996.

[SB92]   L.S. Shapiro and J.M. Brady. Feature-based correspondance: an eigenvector approach. *Image and vision computing*, 10:283–288, 1992.

[SD76]   D. C. Schmidt and L. E. Druffel. A fast backtracking algorithm for test directed graphs for isomorphism. *Journal of the Assoc. for Computing Machinery*, 23:433–445, 1976.

[SYY85]   T. Saito, H. Yamada, and K. Yamamoto. On the data base etl9b of handprinted characters in jis chinese characters and its analysis. *IEICE Trans*, 1985.

[Tay02]   William R. Taylor. Protein structure comparison using bipartite graph matching and its application to protein structure classification. *Molecular and Cellular Proteomics*, pages 334–339, 2002.

[Ull76]   J. R. Ullmann. An algorithm for subgraph isomorphism. *Journal of the Assoc. for Computing Machinery*, 23:433–445, 1976.

[Ume88]   Shinji Umeyama. An eigendecomposition approach to weighted graph matching problems. *Transaction on pattern analysis and machine intelligence*, 10, 1988.

[WH05]   Hong Fang Wang and Edwin R. Hancock. Correspondence matching using kernel principal components analysis and label consistency constraints. *Pattern Recognition*, 2005.

[WKME03] T. Walter, J.-C. Klein, P. Massin, and A. Erignay. Detection of the median axis of vessels in retinal images. *European Journal of Ophthalmology*, 13(2), 2003.

[WMFH04] Yuhang Wang, Fillia Makedon, James Ford, and Heng Huang. A bipartite graph matching framework for finding correspondences between structural elements in two proteins. *Proceedings of the 26th Annual International Conference of the IEEE EMBS*, pages 2972–2975, 2004.