Graph matching with type constraints on nodes and edges*

Catherine Fraikin and Paul Van Dooren

Université catholique de Louvain (UCL), Department of Mathematical Engineering
B-1348 Louvain-La-Neuve, Belgium
fraikin@inma.ucl.ac.be,vdooren@inma.ucl.ac.be

Abstract. In this paper, we consider two particular problems of directed graph matching. The first problem concerns graphs with nodes that have been subdivided into classes of different type. The second problem treats graphs with edges of different types. In the two cases, the matching process is based on a constrained projection of the nodes and of the edges of both graphs in a lower dimensional space. The procedures are formulated as non-convex optimization problems. The objective functions use the adjacency matrices and the constraints on the problem impose the isometry of the so-called projections. Iterative algorithms are proposed to solve the optimization problems. As illustration, we give an example of graph matching for graphs with two types of nodes and graphs with two types of edges.

Keywords. Graph matching, Optimization, Typed nodes, Typed edges

1 Introduction

For many applications in graph theory, a fundamental task is that of finding a correspondence or a measure of similarity between two graphs. This is often referred to as the *graph matching problem*.

Many approaches have been proposed for graph matching and can be classified in two broad classes : the first one tries to find a one-to-one correspondence between some of the vertices of the two graphs (exact graph matching); the second one allows inexact matching and looks for an optimal match even if the considered graphs are structurally different (a survey can be found in [1]). In practice, the second class of methods is the most interesting one because it is more flexible and often gives rise to algorithms that are cheaper to implement.

In this paper we consider first an inexact graph matching method to compare two graphs with nodes that have been subdivided into classes of different type.

* This paper presents research results of the Belgian Network DYSCO (Dynamical Systems, Control, and Optimization), funded by the Interuniversity Attraction Poles Programme, initiated by the Belgian State, Science Policy Office, and a grant Action de Recherche Concertée (ARC) of the Communauté Française de Belgique. The scientific responsibility rests with its authors.

The nodes of the same type in the two graphs are compared to each other, but taking into account the complete interconnection pattern of the graphs. The proposed method is based on the optimization of a certain cost function. The method specializes to the spectral method of Caelli and Kosinov in the case that the graphs to be compared are undirected and contain only one type of nodes [2]. It is also an extension of the method described in [3] which handles the directed graph case for nodes of one type only, which in turn is a low rank approximation of the similarity matrix developed in [4]. The computational technique that we propose is also very similar to that of the above two methods and is essentially a modified power method with special correction applied at each step of the iteration. Since the basic operation to be performed at each step is that of multiplying certain bases with the adjacency matrices of the two graphs, the basic step of the computational procedure can be implemented at low cost for large sparse graphs. An illustration of the matching method is also presented with an application to graphs with nodes that have been classified into two classes. The convergence behavior of the algorithm is also illustrated for this application.

In a second part, the graph matching method is modified in order to compare two directed graphs with edges of different type. This extension is based on a low rank approximation of ideas developed in [5] for one type of edges. The mathematical properties and the computational aspects of this extension are similar to those of the initial problem where the nodes and the edges have no specific label. The method is finally illustrated by comparing two directed graphs for which the edges have been separated into two groups.

2 Notations and definitions

This section first introduces some notations used in the paper and then summarizes some definitions of matrices characterizing the graphs.

2.1 Inner product and gradients

Let $I\!R$ denote the real field and $I\!R^{m\times n}$ denote the set of all $m \times n$ real matrices. X^T represents the transpose of X. For X, $Y \in I\!R^{m\times n}$, the Frobenius *inner product* is defined by

$$\langle X, Y\rangle = \sum_{i=1}^{m}\sum_{j=1}^{n} X_{ij}Y_{ij} = \mathrm{tr}(X^T Y)$$

and its corresponding norm by

$$\|X\| = \langle X, X\rangle^{1/2}.$$

Let $f : I\!R^{m\times n} \to I\!R$ be a differentiable (real-valued) function with matrix argument X. Then the first-order approximation of f at a point X can be expressed as

$$f(X + \Delta) = f(X) + \langle \nabla f(X), \Delta\rangle + o(\|\Delta\|)$$

where the gradient $\nabla f(X)$ is the $m \times n$ matrix whose (i,j) entry is $\frac{\partial f(X)}{\partial X_{i,j}}$. Related to this, we provide some gradients needed in the rest of the paper

$$\nabla \langle A, X^T X \rangle = X(A + A^T), \tag{1}$$
$$\nabla \langle X^T A X, B \rangle = AXB^T + A^T XB. \tag{2}$$

2.2 Adjacency matrix of a graph and extensions

A *graph* G_A consists of a set of *nodes* or *vertices* V_A and a set of *edges* E_A which can be directed or undirected. For a graph with n_A nodes, the $n_A \times n_A$ *adjacency* matrix A of a graph is the matrix whose entry (i,j) is equal to a weight w_{ij} associated to the edge between the nodes i and j.

This "node-node" adjacency matrix is the most common matrix for characterizing the structure of the graph, but other matrices are sometimes used in the literature to describe the graph. In the rest of the section, we present some edge-based matrices introduced in [5]. Let $s_A(i)$ and $t_A(i)$ denote respectively the source node and terminal node of edge i in graph G_A. The number of nodes in G_A is n_A and the number of edges m_A. The adjacency structure of the graph is then described by a pair of $n_A \times m_A$ matrices, the *source-edge matrix* A_S and the *terminus-edge matrix* A_T, defined as follows:

$$[A_S]_{ij} = \begin{cases} \sqrt{w_{it_{A(j)}}} & s_A(j) = i \\ 0 & \text{otherwise} \end{cases}, \tag{3}$$

$$[A_T]_{ij} = \begin{cases} \sqrt{w_{s_{A(j)}i}} & t_A(j) = i \\ 0 & \text{otherwise} \end{cases}. \tag{4}$$

From this representation, it follows that:

- $D_{A_S} = A_S A_S^T$ is a diagonal matrix with the *outgoing capacity* of node i in the ith diagonal entry,
- $D_{A_T} = A_T A_T^T$ is a diagonal matrix with the *ingoing capacity* of node i in the ith diagonal entry,
- $A = A_S A_T^T$ is the classical node-node adjacency matrix.

Moreover, these two matrices A_S and A_T have the advantage of expressing the graph structure from an edge-based point of view, which is interesting with a perspective of labelling the edges in a graph.

3 Graph matching

The matching methods for comparing graphs which have nodes or edges of different types are extensions of the work proposed in [3] where one optimizes a cost function using the adjacency matrices of the two graphs. This section summarizes the procedure and its main properties. Two graphs G_A and G_B of dimensions n_A

and n_B, respectively, were compared by projecting the nodes of the two graphs into a k-dimensional space, where $k \leq \min(n_A, n_B)$ is typically very small (for the purpose of visualization, one chooses e.g. $k = 2, 3$). The projected nodes are obtained as follows: each node i of G_A is mapped to the normalized row i of a matrix $U \in I\!R^{n_A \times k}$, and each node j of G_B is mapped to the normalized row j of $V \in I\!R^{n_B \times k}$, where U and V are matrices optimizing a certain cost function. The projected nodes are thus mapped on the unit sphere in a k-dimensional space, and can then be compared using any preferred matching technique on that manifold (e.g. using nearest-neighbor ideas).

The matrices U and V are obtained from the optimization of a cost function which uses the adjacency matrices A and B of the two graphs. It is given by

$$\max_{U,\,V} \{\langle U^T A U, V^T B V \rangle : U \in \mathcal{Q}_{n_A,k}, V \in \mathcal{Q}_{n_B,k}\} \tag{5}$$

where $\mathcal{Q}_{m,k}$ denotes the set of $m \times k$ matrices with orthonormal columns :

$$\mathcal{Q}_{m,k} = \{X \in I\!R^{m \times k} : \; X^T X = I_k\}.$$

The mathematical properties of the non convex optimization problem (5) are presented in [3]. In general, there is no closed form for the optimal value, except for special matrices A and B (e.g. if A or B is symmetric). For arbitrary matrices, only an upper bound to the problem is obtained. This value is an adequate combination of the eigenvalues of the symmetric and skew-symmetric parts of A and B. An algorithm is also proposed in [3] to solve the optimization problem numerically.

We illustrate the method by comparing the two directed graphs G_A and G_B represented in Figure 1. The two graphs are identical with the exception of the direction of the edge between the nodes 4 and 5. Intuitively, according to the direction of the edges, we expect to associate groups of nodes $1, 2, 3, 4$ of A with $5, 6, 7$ of B and conversely. A normalized solution of problem (5) for $k = 2$ is represented in Figure 3 where the x-axis and the y-axis represent respectively the first and the second columns of U and V. In this projection plane, nodes which are close have the property to be similar. As shown in the Figure, we retrieve the expected grouping of the nodes.

4 Graphs with typed nodes

We will now modify the method described in the previous section for the case of graphs with nodes of different types [6]. The projections U and V will be constrained to have a particular structure such that the nodes of both graphs are only compared with nodes of the same type.

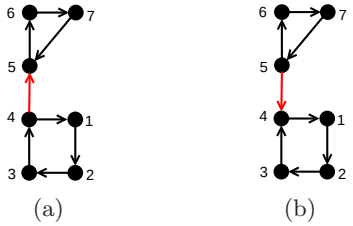


Fig. 1. The two directed graphs A (a) and B (b) that are compared.

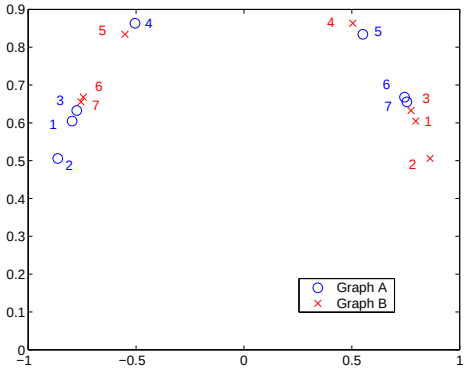


Fig. 2. Projected nodes of Graphs A and B.

4.1 Cost function

If we now have graphs with different types of nodes, we assume that they have been relabelled such that in the corresponding adjacency matrices

$$A = \begin{bmatrix} A_{11} & A_{12} & \dots & A_{1\ell} \\ A_{21} & A_{22} & \dots & A_{2\ell} \\ \vdots & \vdots & \ddots & \vdots \\ A_{\ell 1} & A_{\ell 2} & \dots & A_{\ell\ell} \end{bmatrix}, B = \begin{bmatrix} B_{11} & B_{12} & \dots & B_{1\ell} \\ B_{21} & B_{22} & \dots & B_{2\ell} \\ \vdots & \vdots & \ddots & \vdots \\ B_{\ell 1} & B_{\ell 2} & \dots & B_{\ell\ell} \end{bmatrix},$$

the nodes of the same type $i = 1, \cdots, \ell$ correspond to the same blocks in both matrices A and B. The blocks $A_{i,j} \in I\!R^{n_{A_i} \times n_{A_j}}$ and $B_{i,j} \in I\!R^{n_{B_i} \times n_{B_j}}$ thus describe the edges between nodes of type i to nodes of type j in both A and B.

The rationale behind cost function (5) is that the so-called *projections* U and V describe the dominant behavior of both adjacency matrices A and B, but in terms of a *joint cost function* (5), which emphasizes the *correlation* between both projected matrices. Since our projections U and V should not mix nodes

of different types, we will constrain them to have a block diagonal form :

$$U = \begin{bmatrix} U_1 & 0 & \dots & 0 \\ 0 & U_2 & \dots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & U_\ell \end{bmatrix}, V = \begin{bmatrix} V_1 & 0 & \dots & 0 \\ 0 & V_2 & \dots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & V_\ell \end{bmatrix}, \tag{6}$$

where $U_i \in \mathcal{Q}_{n_{A_i} \times k_i}$ and $V_i \in \mathcal{Q}_{n_{B_i} \times k_i}$, which implies that $k_i \leq \min(n_{A_i}, n_{B_i})$. Notice that this is essentially the same optimization problem as in (5) except for the additional constraint that U and V are block diagonal *projections*, which of course prevents the mixing of types in the projected nodes. We will subsequently have to match nodes within each type i, using again a preferred matching algorithm, on a sphere in a k_i-dimensional space, and this for $i = 1, \cdots, \ell$. The mapping of the projected nodes on a sphere is again obtained by a row normalization of U and V. In the rest of this paper, we will assume for ease of notation that $\ell = 2$, i.e. that there are only two different types of nodes. All results extend trivially to the case of arbitrary number of types ℓ.

The mathematical properties of the optimization problem (5) were presented in [3] for $\ell = 1$. We give here a very short proof for the extension to the constrained case with $\ell = 2$ blocks. We thus consider two graphs with partitioned adjacency matrices

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}, \quad B = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

with $A_{i,j} \in I\!R^{n_{A_i} \times n_{A_j}}$ and $B_{i,j} \in I\!R^{n_{B_i} \times n_{B_j}}$ and orthogonal matrices $U_i \in \mathcal{Q}_{n_{A_i} \times k_i}$ and $V_i \in \mathcal{Q}_{n_{B_i} \times k_i}$, for $i = 1, 2$ and $j = 1, 2$. The optimization problem (5) then becomes

$$\max_{U_i, V_i} \{ \langle \begin{bmatrix} U_1^T A_{11} U_1 & U_1^T A_{12} U_2 \\ U_2^T A_{21} U_1 & U_2^T A_{22} U_2 \end{bmatrix}, \begin{bmatrix} V_1^T B_{11} V_1 & V_1^T B_{12} V_2 \\ V_2^T B_{21} V_1 & V_2^T B_{22} V_2 \end{bmatrix} \rangle : U_i \in \mathcal{Q}_{n_{A_i}, k_i}, V_i \in \mathcal{Q}_{n_{B_i}, k_i}, i = 1, 2 \} \tag{7}$$

which is a continuous function on a compact domain. Therefore, there always exists an optimal solution $(U_i, V_i), i = 1, 2$ such that the first order conditions are satisfied. By using some of the trace properties[1], the objective function becomes

$$\begin{aligned} &\langle U_1^T A_{11} U_1, V_1^T B_{11} V_1 \rangle + \langle U_2^T A_{22} U_2, V_2^T B_{22} V_2 \rangle + \\ &\langle U_1^T A_{12} U_2, V_1^T B_{12} V_2 \rangle + \langle U_2^T A_{21} U_1, V_2^T B_{21} V_1 \rangle. \end{aligned} \tag{8}$$

The first-order derivative conditions can be derived from the Lagrangian:

$$\begin{aligned} L(U_1, U_2, V_1, V_2, X_1, X_2, Y_1, Y_2) = &\langle U_1^T A_{11} U_1, V_1^T B_{11} V_1 \rangle + \langle U_1^T A_{12} U_2, V_1^T B_{12} V_2 \rangle \\ &+ \langle U_2^T A_{21} U_1, V_2^T B_{21} V_1 \rangle + \langle U_2^T A_{22} U_2, V_2^T B_{22} V_2 \rangle \\ &+ \langle X_1, (I - U_1^T U_1) \rangle + \langle X_2, (I - U_2^T U_2) \rangle \\ &+ \langle Y_1, (I - V_1^T V_1) \rangle + \langle Y_2, (I - V_2^T V_2) \rangle \end{aligned}$$

[1] $trA = trA^T, \quad trAB = trBA, \quad tr(A + B) = trA + trB.$

where X_i and Y_i are symmetric matrices of Lagrange multipliers for the orthogonality constraints. By setting the partial gradients of this Lagrangian to zero (using (1) and (2)), the first order conditions are found to be :

$$\begin{aligned}
U_1X_1 &= [A_{11}U_1V_1^TB_{11}^T + A_{11}^TU_1V_1^TB_{11} + A_{12}U_2V_2^TB_{12}^T + A_{21}^TU_2V_2^TB_{21}]V_1,\\
V_1Y_1 &= [B_{11}V_1U_1^TA_{11}^T + B_{11}^TV_1U_1^TA_{11} + B_{12}V_2U_2^TA_{12}^T + B_{21}^TV_2U_2^TA_{21}]U_1,\\
U_2X_2 &= [A_{22}U_2V_2^TB_{22}^T + A_{22}^TU_2V_2^TB_{22} + A_{21}U_1V_1^TB_{21}^T + A_{12}^TU_1V_1^TB_{12}]V_2,\\
V_2Y_2 &= [B_{22}V_2U_2^TA_{22}^T + B_{22}^TV_2U_2^TA_{22} + B_{21}V_1U_1^TA_{21}^T + B_{12}^TV_1U_1^TA_{12}]U_2,
\end{aligned}$$

where $X_i = Y_i$ are symmetric matrices. Clearly the cost in (7) does not change if one multiplies each pair U_i, V_i by a common $k_i \times k_i$ orthogonal matrix Q_i. That degree of freedom can also be used to impose that $X_i = Y_i$ are also equal to diagonal matrices Λ_i (see [3] for more details).

4.2 Computational aspects

In this section we present an iterative algorithm to find a critical point of (7). It is a simple recursive algorithm based on the Singular Value Decomposition, which provides interesting results. A convergence proof is still missing but numerical experiments always show linear convergence to an equilibrium point, provided the shifts $s_i, i = 1, 2$ are chosen appropriately.

The proposed iterative algorithm to compute a critical point of (7) is as follows

$$\begin{aligned}
U_{1+}\Sigma_{1+}V_{1+}^T + U_{1-}\Sigma_{1-}V_{1-}^T &= A_{11}U_1V_1^TB_{11}^T + A_{11}^TU_1V_1^TB_{11}\\
&\quad + A_{12}U_2V_2^TB_{12}^T + A_{21}^TU_2V_2^TB_{21} + s_1U_1V_1^T \quad (9)\\
U_{2+}\Sigma_{2+}V_{2+}^T + U_{2-}\Sigma_{2-}V_{2-}^T &= A_{22}U_2V_2^TB_{22}^T + A_{22}^TU_2V_2^TB_{22}\\
&\quad + A_{21}U_1V_1^TB_{21}^T + A_{12}^TU_1V_1^TB_{12} + s_2U_2V_2^T \quad (10)
\end{aligned}$$

where U_{i+} and V_{i+} are orthogonal complements of U_{i-} and V_{i-}. The scalars s_i are positive numbers sufficiently large (see [3] for more details) and $\Sigma_{i\pm}$ are diagonal matrices. The updating of the matrices U_i, V_i is done at each iteration according to

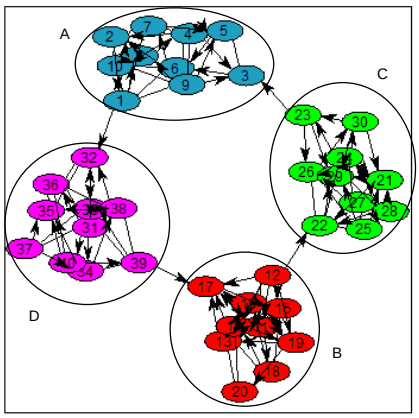
$$U_1 := U_{1+}, \quad U_2 := U_{2+}, \quad V_1 := V_{1+}, \quad V_2 := V_{2+}. \tag{11}$$

It is easy to see that $U_{1+}\Sigma_{1+}V_{1+}^T$ and $U_{2+}\Sigma_{2+}V_{2+}^T$ are the best rank k_i approximations of the respective right hand sides. In practice the previous iteration is realized by the application of the SVD algorithm to the right-hand side of the equation, which is a matrix of rank at most $5k_i$. This can be exploited of course in the application of the SVD algorithm to these right hand sides. For sparse matrices A and B, the complexity of the computation of one iteration step (9), (10), (11) is then only linear in n_A and n_B.

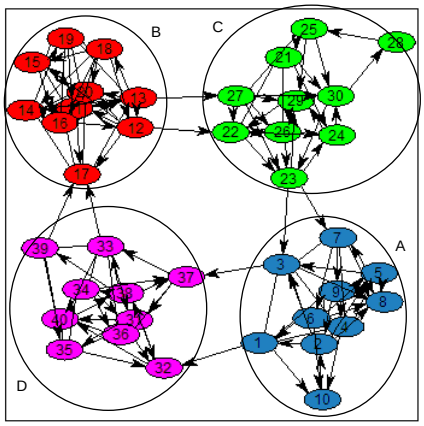
One can show that the critical points of the cost function (7) are fixed points of the iteration (9), (10), (11) and vice-versa. The proof of this is very similar to that of the one block case [3] and is omitted here.

4.3 Experiments in graph matching

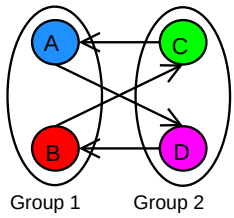
In order to apply this problem to graph matching, we use it to compare two directed graphs, the nodes of which have been divided into two types, labelled group 1 and 2.



(a) Graph A with four groups of 10 random nodes.



(b) Graph B with four groups of 10 random nodes.

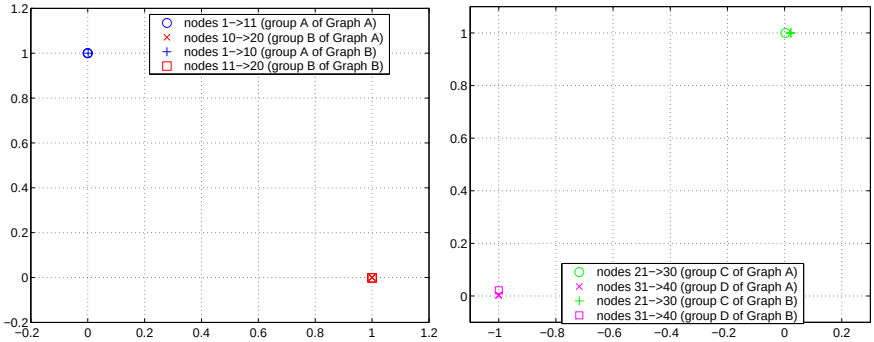


(c) Bipartite partition.

Fig. 3. Two *essentially* bipartite graphs.

In Figure 3 we show two graphs that are *essentially* bipartite in the sense that nodes from clusters A and B (denoted as group 1 in Figure 3(c)) point to nodes from clusters C and D (denoted as group 2 in Figure 3(c)), and vice versa. These are not *true* bipartite graphs because within the clusters, there are random connections between the nodes of that cluster. But clearly one hopes to detect a close connection between the groups of nodes of type A and B in both graphs and those of type C and D in both graphs. Indeed, when imposing the constraint that only nodes of group 1 and 2 can be compared with each other in both graphs, then there is a clear distinction between the subgroups of type A and B

in group 1, on the one hand, and those of type C and D in group 2, on the other hand. This is also what is observed in Figure 4 when projecting both groups of nodes in a two dimensional space.



(a) Projected nodes of Group 1 of both graphs.

(b) Projected nodes of Group 2 of both graphs.

Fig. 4. Closeness between nodes of the same clusters in each group.

When one zooms in on each of the four clusters in Graphs A and B, one clearly sees in Figure 5 that the nodes of these clusters are different, but quite close to each other.

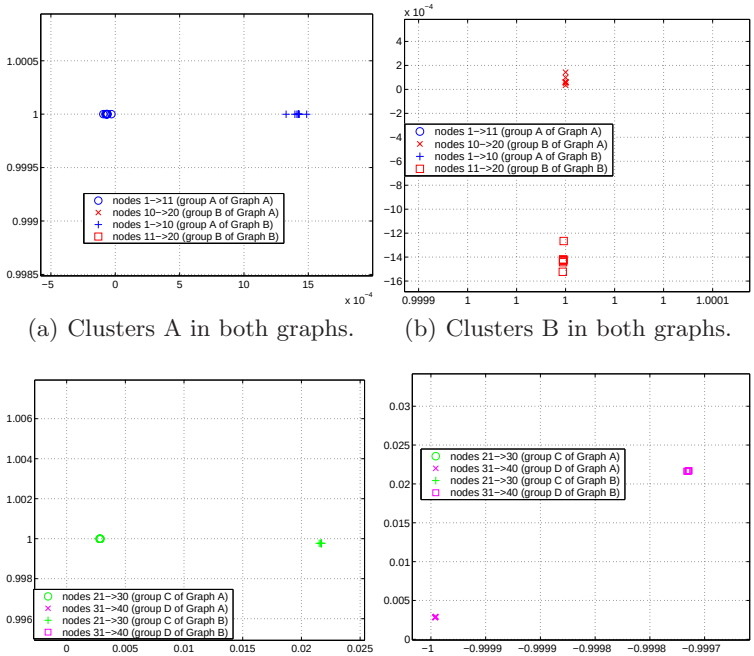
We also show the convergence behavior of our algorithm for this example. We measured the convergence by checking the maximal residual norm at each step

$$Residual(step) := \max(\|U_1 V_1^T - U_{1+} V_{1+}^T\|_2, \|U_2 V_2^T - U_{2+} V_{2+}^T\|_2) \qquad (12)$$

which is what we plot in Figure 6(a) below. One can clearly observe linear convergence of our algorithm. Finally, we show in Figure 6(b) the convergence of the actual value of the cost, as a function of the iteration step.

Let us remark that we have presented one solution corresponding to a local minima of problem (7). For this solution, the nodes of cluster A in Graph A are associated to nodes of cluster A in Graph B, and so on for the four clusters of nodes. For another local minimum or solution, the nodes of cluster A in Graph A would have been associated to nodes of cluster B in Graph B, the nodes of cluster B in Graph A to nodes of cluster A in Graph B, the nodes of cluster C in Graph A to nodes of cluster D in Graph B and finally, the nodes of cluster D in Graph A would have been associated to nodes of cluster C in Graph B. Depending on the initial condition for the iteration, the solution will be different, but the nodes will always be associated with nodes of the same type.

If we suppress the types in the previous example, clusters of nodes will now be compared without any constraints of type. One then automatically obtains



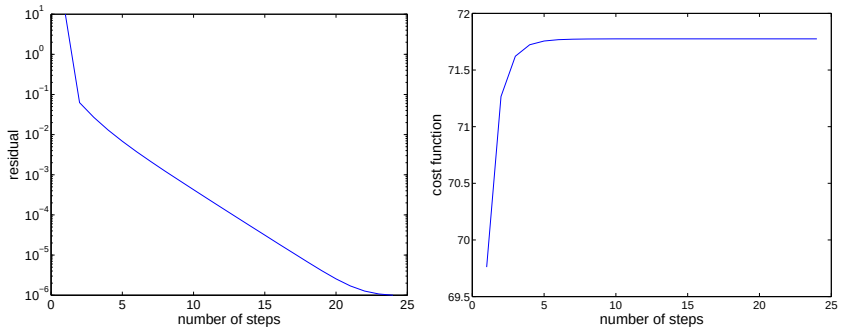
(a) Clusters A in both graphs.

(b) Clusters B in both graphs.

(c) Clusters C in both graphs.

(d) Clusters D in both graphs.

Fig. 5. Zooming in on the four different groups in Graphs A and B.



(a) Linear convergence of residual. (b) Convergence of the cost.

Fig. 6. Convergence behavior.

four local minima, that correspond to cyclic permutations of the clusters A,B,C and D in Graphs A and B.

5 Graphs with typed edges

An other extension of the general method concerns graphs with edges of different types. The method will now rely on the projection of the nodes and of the edges of both graphs in a lower-dimensional space. For graphs with different types of edges, the projected edges will be constrained to a particular structure.

5.1 Coupled node-edge projections

Up to now, we have introduced a notion of projection of the nodes of two graphs, which is then used to compare the graphs. In order to take into account edge types, we need to introduce also a measure of similarity between the edges of both graphs. This measure will be based, similarly to what we did for the nodes, on a projection of the edges in a lower dimensional space.

The matching process is based on the coupled projection of the nodes and of the edges of the two graphs in a lower-dimensional subspace by the optimization of a certain cost function. The method is a low rank variant of the method described in [5] and an extension of the problem presented in Section 3. The number of nodes in G_A (resp. G_B) is denoted by n_A (n_B) and the number of edges by m_A (m_B). The two graphs are compared by projecting the nodes and the edges into a k_1 (resp. k_2)-dimensional space, where $k_1 \leq \min(n_A, n_B)$ and $k_2 \leq \min(m_A, m_B)$ are typically small. The projections of the nodes are represented by $U_1 \in I\!R^{n_A \times k_1}$ and $V_1 \in I\!R^{n_B \times k_1}$ and the projections of the edges by $U_2 \in I\!R^{m_A \times k_2}$ and $V_2 \in I\!R^{m_B \times k_2}$. The comparison of the graph uses the projections as explained in Section 3.

The matrices U_i and V_i, $i = 1, 2$ are obtained from the optimization of a cost function which uses the source- and terminus-edge matrices of the two graphs. It corresponds to

$$\max_{U_1, V_1, U_2, V_2} \{\langle U_1^T A_S U_2, V_1^T B_S V_2\rangle + \langle U_1^T A_T U_2, V_1^T B_T V_2\rangle : \\ U_1 \in \mathcal{Q}_{m_A,k_1}, V_1 \in \mathcal{Q}_{m_B,k_1}, U_2 \in \mathcal{Q}_{n_A,k_2}, V_2 \in \mathcal{Q}_{n_B,k_2}\} \quad (13)$$

which is a continuous function on a compact domain. Therefore, there always exists a solution (U_i, V_i), $i = 1, 2$ such that the first-order conditions are satisfied. These conditions are derived from the Lagrangian $L(U_1, V_1, U_2, V_2, X_1, Y_1, X_2, Y_2)$:

$$\begin{aligned} L(U_1, V_1, U_2, V_2, X_1, Y_1, X_2, Y_2) = {} & \langle U_1^T A_S U_2, V_1^T B_S V_2\rangle + \langle U_1^T A_T U_2, V_1^T B_T V_2\rangle \\ & + \langle X_1, (I - U_1^T U_1)\rangle + \langle Y_1, (I - V_1^T V_1)\rangle \\ & + \langle X_2, (I - U_2^T U_2)\rangle + \langle Y_2, (I - V_2^T V_2)\rangle \end{aligned}$$

where X_i and Y_i are symmetric matrices of Lagrange multipliers for the orthogonality constraints. By setting the partial gradients of this Lagrangian to zero, the first order conditions are found to be :

$$\begin{aligned} U_1 X_1 &= \left[A_S U_2 V_2^T B_S^T + A_T U_2 V_2^T B_T^T\right] V_1, \\ V_1 Y_1 &= \left[B_S V_2 U_2^T A_S^T + B_T V_2 U_2^T A_T^T\right] U_1, \\ U_2 X_2 &= \left[A_S^T U_1 V_1^T B_S + A_T^T U_1 V_1^T B_T\right] V_2, \\ V_2 Y_2 &= \left[B_S^T V_1 U_1^T A_S + B_T^T V_1 U_1^T A_T\right] U_2, \end{aligned} \quad (14)$$

where $X_i = Y_i$, $i = 1, 2$ are symmetric matrices. One can easily see that the cost in (13) does not change by multiplying each pair U_i, V_i, $i = 1, 2$ by a common $k_i \times k_i$ orthogonal matrix. That degree of freedom can be used to choose $X_i = Y_i$ diagonal.

5.2 Edges of different types

If we now have graphs with ℓ different types of edges, the previous problem can be extended to take into account this particularity and to match together only edges of the same type. We assume that the edges have been labelled such that in the corresponding source- and terminus-edge matrices

$$A_S = \left[\, A_{S_1}\ A_{S_2}\ \cdots\ A_{S_\ell} \,\right], \quad A_T = \left[\, A_{T_1}\ A_{T_2}\ \cdots\ A_{T_\ell} \,\right],$$

$$B_S = \left[\, B_{S_1}\ B_{S_2}\ \cdots\ B_{S_\ell} \,\right], \quad B_T = \left[\, B_{T_1}\ B_{T_2}\ \cdots\ B_{T_\ell} \,\right],$$

the edges of the same type $i = 1, 2, \cdots, \ell$ correspond to the same blocks A_{S_i} (A_{T_i}), B_{S_i} (B_{T_i}) in both A_S (A_T) and B_S (B_T). The blocks $A_{S_i}(A_{T_i}) \in I\!R^{n_A \times m_{A_i}}$ and $B_{S_i}(B_{T_i}) \in I\!R^{n_B \times m_{B_i}}$, $i = 1, 2, \cdots, \ell$ thus correspond to edges of the same type, where m_{A_i} and m_{B_i}, $i = 1, 2$ represent the number of edges of type i in G_A and G_B respectively.

Since the projections of the edges U_2 and V_2 in (13) should not mix edges of different types, they will be constrained to have a block diagonal form:

$$U_2 = \begin{bmatrix} U_{21} & O & \cdots & 0 \\ 0 & U_{22} & \cdots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & U_{2\ell} \end{bmatrix}, \qquad V_2 = \begin{bmatrix} V_{21} & O & \cdots & 0 \\ 0 & V_{22} & \cdots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & V_{2\ell} \end{bmatrix} \tag{15}$$

where U_{2i} has dimension $m_{A_i} \times k_i$ and $U_{2i}^T U_{2i} = I_{k_i}$, V_{2i} has dimension $m_{B_i} \times k_i$ and $V_{2i}^T V_{2i} = I_{k_i}$, $i = 1, 2, \cdots, \ell$.

In the following, we will assume that the nodes are only of two types, i.e. $\ell = 2$ and that $k_i = k$, $i = 1, 2$. All results extend to the case of arbitrary number of types ℓ.

The optimization problem (13) becomes

$$\begin{aligned} \max_{\substack{U_1, V_1, \\ U_{21}, V_{21}, \\ U_{22}, V_{22}}} \; & \{\langle U_1^T A_{S1} U_{21}, V_1^T B_{S1} V_{21}\rangle + \langle U_1^T A_{S2} U_{22}, V_1^T B_{S2} V_{22}\rangle \\ & + \langle U_1^T A_{T1} U_{21}, V_1^T B_{T1} V_{21}\rangle + \langle U_1^T A_{T2} U_{22}, V_1^T B_{T2} V_{22}\rangle : \\ & U_1 \in \mathcal{Q}_{n_A,k}, V_1 \in \mathcal{Q}_{n_B,k}, U_{2i} \in \mathcal{Q}_{m_{A_i},k}, V_{2i} \in \mathcal{Q}_{m_{B_i},k}, i = 1, 2\}. \end{aligned} \tag{16}$$

The first-order conditions can be derived from the Lagrangian:

$$\begin{aligned} L(U_1, V_1, U_{21}, U_{22}, V_{21}, V_{22}, X_1, X_2, X_3, Y_1, Y_2, Y_3) = \\ & \langle U_1^T A_{S1} U_{21}, V_1^T B_{S1} V_{21}\rangle + \langle U_1^T A_{S2} U_{22}, V_1^T B_{S2} V_{22}\rangle \\ & + \langle U_1^T A_{T1} U_{21}, V_1^T B_{T1} V_{21}\rangle + \langle U_1^T A_{T2} U_{22}, V_1^T B_{T2} V_{22}\rangle \\ & + \langle X_1, I - U_1^T U_1\rangle + \langle Y_1, I - V_1^T V_1\rangle + \langle X_2, I - U_{21}^T U_{21}\rangle \\ & + \langle X_3, I - U_{22}^T U_{22}\rangle + \langle Y_2, I - V_{21}^T V_{21}\rangle + \langle Y_3, I - V_{22}^T V_{22}\rangle \end{aligned}$$

where X_i and Y_i are symmetric matrices of Lagrange multipliers for the orthogonality constraints. By setting the partial gradients of this Lagrangian to zero, the first order conditions are found to be :

$$\begin{aligned} U_1 X_1 &= \left[A_{S_1} U_{21} V_{21}^T B_{S_1}^T + A_{T_1} U_{21} V_{21}^T B_{T_1}^T + A_{S_2} U_{22} V_{22}^T B_{S_2}^T + A_{T_2} U_{22} V_{22}^T B_{T_2}^T\right] V_1, \\ V_1 Y_1 &= \left[B_{S_1} V_{21} U_{21}^T A_{S_1}^T + B_{T_1} V_{21} U_{21}^T A_{T_1}^T + B_{S_2} V_{22} U_{22}^T A_{S_2}^T + B_{T_2} V_{22} U_{22}^T A_{T_2}^T\right] U_1, \\ U_{21} X_2 &= \left[A_{S_1}^T U_1 V_1^T B_{S_1} + A_{T_1}^T U_1 V_1^T B_{T_1}\right] V_{21}, \\ V_{21} Y_2 &= \left[B_{S_1}^T V_1 U_1^T A_{S_1} + B_{T_1}^T V_1 U_1^T A_{T_1}\right] U_{21}, \\ U_{22} X_3 &= \left[A_{S_2}^T U_1 V_1^T B_{S_2} + A_{T_2}^T U_1 V_1^T B_{T_2}\right] V_{22}, \\ V_{22} Y_3 &= \left[B_{S_2}^T V_1 U_1^T A_{S_2} + B_{T_2}^T V_1 U_1^T A_{T_2}\right] U_{22} \end{aligned}$$

where $X_i = Y_i$, $i = 1, 2, 3$ are symmetric matrices which can be chosen diagonal without affecting the value of the objective function.

5.3 Computational aspects

An iterative algorithm to compute a critical point of (16), which in fact reduces to (13) for $\ell = 1$, is very similar to the algorithm presented in Section 4.2. This recursive algorithm is once again based on the SVD. The convergence has not yet been proved but numerical experiments show linear convergence to an equilibrium point, provided the shifts s_i, $i = 1, 2, 3$ are chosen appropriately.

The proposed iterative algorithm is the following:

$$\begin{aligned} U_{1+}\Sigma_{1+}V_{1+}^T + U_{1-}\Sigma_{1-}V_{1-}^T &= A_{S_1}U_{21}V_{21}^T B_{S_1}^T + A_{T_1}U_{21}V_{21}^T B_{T_1}^T \\ &\quad + A_{S_2}U_{22}V_{22}^T B_{S_2}^T + A_{T_2}U_{22}V_{22}^T B_{T_2}^T + s_1 U_1 V_1^T \quad (17) \\ U_{21+}\Sigma_{2+}V_{21+}^T + U_{21-}\Sigma_{2-}V_{21-}^T &= A_{S_1}^T U_1 V_1^T B_{S_1} + A_{T_1}^T U_1 V_1^T B_{T_1} + s_2 U_{21}V_{21}^T \quad (18) \\ U_{22+}\Sigma_{3+}V_{22+}^T + U_{22-}\Sigma_{3-}V_{22-}^T &= A_{S_2}^T U_1 V_1^T B_{S_2} + A_{T_2}^T U_1 V_1^T B_{T_2} + s_3 U_{22}V_{22}^T \quad (19) \end{aligned}$$

where U_{x+} and V_{x+} are orthogonal complements of U_{x-} and V_{x-}. The scalars s_i are positive numbers sufficiently large and $\Sigma_{i\pm}$ are diagonal matrices. The updating of the matrices is done at each iteration according to

$$\begin{aligned} U_1 := U_{1+}, \quad U_{21} := U_{21+}, \quad U_{22} := U_{22+}, \\ V_1 := V_{1+}, \quad V_{21} := V_{21+}, \quad V_{22} := V_{22+}. \end{aligned} \quad (20)$$

It is easy to see that $U_{1+}\Sigma_{1+}V_{1+}^T$, $U_{21+}\Sigma_{2+}V_{21+}^T$ and $U_{22+}\Sigma_{3+}V_{22+}^T$ are the best rank k_i approximations of the respective right hand sides. In practice the previous iteration is realized by the application of the SVD algorithm to the right-hand side of the equation, which is a matrix of rank at most $5k_i$. This can of course be exploited in the application of the SVD algorithm to these right hand sides.

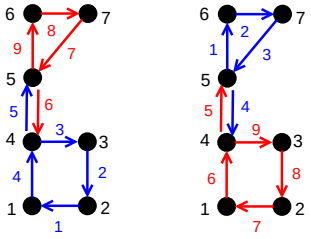
One can show that the critical points of the cost function (16) are fixed points of the iteration (17), (18), (19), (20) and vice-versa. The proof of this is very similar to that of the one block case [3] and is omitted here.

5.4 Examples

As illustration of the method, we compare two directed graphs represented in Figure 7, the edges of which have been divided into two types, the red and the blue ones. We apply the algorithm of Section 5.3 which provides the projections of the nodes of the two graphs U_1 and V_1. The normalized projected nodes are plotted in Figure 8. The nodes are correctly grouped according to the type of their source and terminus edges. This solution represents a particular local minimum of the cost function.

6 Conclusion

In this paper, we extend the projected correlation method described in [3], which can be used to perform graph matching between two graphs represented by their adjacency matrices A and B.



(a) Graph A (b) Graph B

Fig. 7. Graphs to be compared.

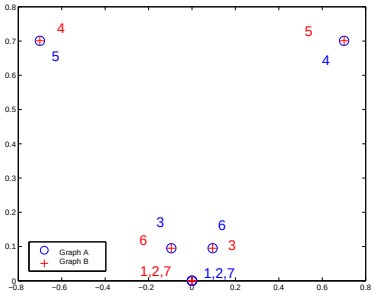


Fig. 8. Projected nodes of Graphs A and B

The first extension concerns the graphs with nodes of different types. Such graphs represent e.g. molecules where the nodes represent atoms of different types (say C, O, N, ...). The proposed extension constrains the matching such that the nodes of both graphs can only be compared when their types are the same : both graphs are thus assumed to be partitioned in subgraphs of equal type. The proposed cost function nevertheless uses the connectivity between nodes of different type in the global cost, to be minimized.

The second extension of the general method concerns graphs with edges of different types. An example of such a graph is a social graph, i.e. a set of people with some pattern of interactions between them. In such a graph, the nodes represent e.g. the individuals and the edges the relationships between them (friendships, business relationships, family ties, ...). The method will now rely on the projection of the nodes and of the edges of both graphs in a lower-dimensional space. For graphs with different types of edges, the projected edges will be constrained to a particular structure.

The case of graphs with different types of nodes and edges could easily be treated by combining both approaches.

References

1. Conte, D., Foggia, P., Sansone, C., Vento, M.: Thirty years of graph matching in pattern recognition. Int. J. Pattern Recognition and Artificial Intelligence **18** (2004) 265–298
2. Caelli, T., Kosinov, S.: An eigenspace projection clustering method for inexact graph matching. IEEE Trans. Pattern Analysis and Machine Intelligence **26** (2004) 515–519
3. Fraikin, C., Nesterov, Y., Dooren, P.V.: Optimizing the coupling between two isometric projections of matrices. Submitted to SIAM J. Matrix Anal. Appl. (2006)
4. Blondel, V.D., Gajardo, A., Heymans, M., Senellart, P., Van Dooren, P.: A measure of similarity between graph vertices: applications to synonym extraction and Web searching. SIAM Rev. **46** (2004) 647–666
5. Zager, L., Verghese, G.: Graph similarity scoring and matching. To appear in Applied Math. Letters (2007)
6. Fraikin, C., Dooren, P.V.: Graph matching with type constraints. Proc. Europ. Control Conf.(ECC'07) (2007)