A Randomized Heuristic for Scene Recognition by Graph Matching

Maria C. Boeres[1], Celso C. Ribeiro[2], and Isabelle Bloch[3]

[1] Universidade Federal do Espírito Santo, Department of Computer Science, R. Fernando Ferrari, Campus de Goiabeiras, Vitória, ES 29060-970, Brazil.
boeres@inf.ufes.br

[2] Universidade Federal Fluminense, Department of Computer Science, Rua Passo da Pátria 156, Niterói, RJ 24210-240, Brazil.
celso@inf.puc-rio.br

[3] Ecole Nationale Supérieure des Télécommunications, CNRS URA 820, 46 rue Barrault, 75634 Paris Cedex 13, France.
Isabelle.Bloch@enst.fr

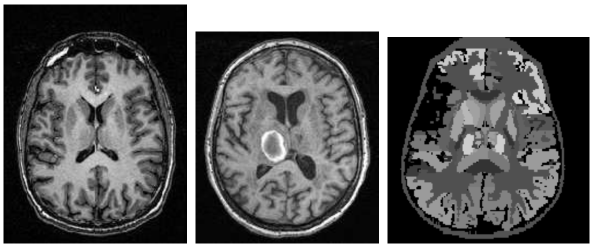
Abstract. We propose a new strategy for solving the non-bijective graph matching problem in model-based pattern recognition. The search for the best correspondence between a model and an over-segmented image is formulated as a combinatorial optimization problem, defined by the relational attributed graphs representing the model and the image where recognition has to be performed, together with the node and edge similarities between them. A randomized construction algorithm is proposed to build feasible solutions to the problem. Two neighborhood structures and a local search procedure for solution improvement are also proposed. Computational results are presented and discussed, illustrating the effectiveness of the combined approach involving randomized construction and local search.

1 Introduction

The recognition and the understanding of complex scenes require not only a detailed description of the objects involved, but also of the spatial relationships between them. Indeed, the diversity of the forms of the same object in different instantiations of a scene, and also the similarities of different objects in the same scene, make relationships between objects of prime importance in order to disambiguate the recognition of objects with similar appearance. Graph based representations are often used for scene representation in image processing [6, 9, 11, 20, 21]. Vertices of the graphs usually represent the objects in the scenes, while their edges represent the relationships between the objects. Relevant information for the recognition is extracted from the scene and represented by relational attributed graphs. In model-based recognition, both the model and the scene are represented by graphs.

The assumption of a bijection between the elements in two instantiations of the same scene is too strong for many problems. Usually, the model has a schematic aspect. Moreover, the construction of the image graph often relies on segmentation techniques that may fail in accurately segmenting the image into meaningful entities. Therefore, no isomorphism can be expected between both graphs and, in consequence, scene recognition may be better expressed as an non-bijective graph matching problem.

Our motivation comes from an application in medical imaging, in which the goal consists in recognizing brain structures from 3D magnetic resonance images, previously processed by a segmentation method. The model consists of an anatomical atlas. A graph is built from the atlas, in which each node represents exactly one anatomical structure of interest. Edges of this graph represent spatial relationships between the anatomical structures. Inaccuracies constitute one of the main characteristics of the problem. Objects in the image are segmented and all difficulties with object segmentation will be reflected in the representation, such as over-segmentation, unexpected objects found in the scene (pathologies for instance), expected objects not found and deformations of objects [13]. Also, the attributes computed for the image and the model may be imprecise. To illustrate these difficulties, Figure 1 presents slices of three different volumes: (a) a normal brain, (b) a pathological brain with a tumor, and (c) the representation of a brain atlas where each grey level corresponds to a unique connected structure. Middle dark structures (lateral ventricles) are much bigger in (b) than in (a). The white hyper-signal structure (tumor) does not appear in the atlas (c) nor in the normal brain (a). Similar problems occur in other applications, such as aerial or satellite image interpretation using a map, face recognition, and character recognition.



(a) (b) (c)

Fig. 1. Examples of magnetic resonance images: (a) axial slice of a normal brain, (b) axial slice of a pathological brain with a tumor, and (c) axial slice of a brain atlas.

This paper focuses on algorithms for the non-bijective graph matching problem [1, 7, 10, 13, 15, 17, 19], which is defined by the relational attributed graphs representing the model and the over-segmented image, together with the node and edge similarities between their nodes and edges. Section 2 describes our formulation of the search for the best correspondence between the two graphs as a non-bijective graph matching problem. We discuss the nature of the objective function and of the constraints of the graph matching problem. A randomized construction algorithm is proposed in Section 3 to build feasible solutions. Besides the quality of the solutions found, this algorithm may also be used as a robust generator of initial solutions for a GRASP metaheuristic [16] or for population methods such as the genetic algorithm described in [14]. A local search algorithm is proposed in Section 4 to improve the solutions obtained by the construction algorithm. Numerical results obtained with the randomized construction and the local search algorithms are presented and discussed in Section 5. They illustrate the robustness of the construction algorithm and the improvements attained by the local search algorithm in terms of solution quality and object identification. Concluding remarks are presented in the last section.

2 Non-bijective Graph Matching

Attributed graphs are widely used in pattern recognition problems. The definition of the attributes and the computation of their values are specific to each application and problem instance. Fuzzy attributed graphs are used for recognition under imprecisions [2–5, 12–15]. The construction of a fuzzy attributed graph depends on the imperfections of the scene or of the reference model, and on the attributes of the object relations. The common point is that there is always a single vertex for each region of each image. Differences may occur due to the strategy applied for the creation of the edge set, as a result of the chosen attributes or of defects in scene segmentation. Once the graph is built, the next step consists in computing the attributes of vertices and edges. Finally, vertex-similarity and edge-similarity matrices are computed from the values of the attributed graphs, relating each pair of vertices and each pair of edges, one of them from the model and the other from the segmented image.

Two graphs are used to represent the problem: $G_1 = (N_1, E_1)$ represents the model, while $G_2 = (N_2, E_2)$ represents the over-segmented image. In each case, N_i denotes the vertex set and E_i denotes the edge set, $i \in \{1, 2\}$. We assume that $|N_1| \leq |N_2|$, which is the case when the image is over-segmented with respect to the model.

A solution to the non-bijective graph matching problem is defined by a set of associations between the nodes of G_1 and G_2. Each node of G_2 is associated with one node of G_1. These assignments are represented by binary variables: $x_{ij} = 1$ if nodes $i \in N_1$ and $j \in N_2$ are associated, $x_{ij} = 0$ otherwise. The set $A(i) = \{j \in N_2 \, | x_{ij} = 1\}$ denotes the subset of vertices of N_2 associated with vertex $i \in N_1$. To ensure that the structure of G_1 appears within G_2, we favor solutions where a correspondence between edges also implies a correspondence between their extremities (edge association condition). Thus, edge associations are derived from vertex associations, according to the following rule: edge $(a, b) \in E_1$ is associated with all edges $(a', b') \in E_2$ such that (i) $a' \in N_2$ is associated with $a \in N_1$ and $b' \in N_2$ is associated with $b \in N_1$ or (ii) $a' \in N_2$ is associated with $b \in N_1$ and $b' \in N_2$ is associated with $a \in N_1$.

A good matching is a solution in which the associations correspond to high similarity values. Similarity matrices are constructed from similarity values calculated from graph attributes. The choice of these attributes depends on the images. Let S^v (resp. S^e) denote an $|N_1| \times |N_2|$ (resp. $|E_1| \times |E_2|$) vertex-similarity (resp. edge-similarity) matrix, where the elements $s^v(i,j)$ (resp. $s^e((i,i'),(j,j'))$) $\in [0,1]$ represent the similarity between vertices (resp. edges) $i \in N_1$ and $j \in N_2$ (resp. $(i,i') \in E_1$ and $(j,j') \in E_2$). The value of any solution is expressed by an objective function, defined for each solution x as

$$f(x) = \frac{\alpha}{|N_1| \cdot |N_2|} f^v(x) + \frac{(1-\alpha)}{|E_1| \cdot |E_2|} f^e(x),$$

with

$$f^v(x) = \sum_{i \in N_1} \sum_{j \in N_2} (1 - |x_{ij} - s^v(i,j)|)$$

and

$$f^e(x) = \sum_{(i,i') \in E_1} \sum_{(j,j') \in E_2} (1 - |\max\{x_{ij}x_{i'j'}, x_{ij'}x_{i'j}\} - s^e((i,i'),(j,j'))|),$$

where α is a parameter used to weight each term of f. This function consists of two terms which represent the vertex and edge contributions to the measure of the solution quality associated with each correspondence. Vertex and edge associations with high similarity values are privileged, while those with low similarity values are penalized. The first term represents the average vertex contribution to the correspondence. The second term represents the average edge contribution to the correspondence and acts to enforce the edge association condition. For instance, if $s^e((i,i'),(j,j'))$ is high and there are associations between the extremities of edges (i,i') and (j,j'), then $\max\{x_{ij}x_{i'j'}, x_{ij'}x_{i'j}\} = 1$ and the edge contribution is high. On the contrary, if the extremities of edges (i,i') and (j,j') are not associated, then $\max\{x_{ij}x_{i'j'}, x_{ij'}x_{i'j}\} = 0$ and the edge contribution is null. This function behaves appropriately when the image features are well described by the graph attributes.

The search is restricted only to solutions in which each vertex of N_2 has to be associated with exactly one vertex of N_1. The rationale for this condition is that image segmentation is performed by an appropriate algorithm which preserves the boundaries and, in consequence, avoids situations in which one region of the segmented image is located in the frontier of two adjacent regions of the model: Constraint (1): For every $j \in N_2$, there exists exactly one node $i \in N_1$ such that $x_{ij} = 1$, i.e. $|A^{-1}(j)| = 1$.

The quality of the input data (vertex and edge similarity matrices) is primordial for the identification of the best correspondence. However, as this quality is not always easy to be achieved in real applications, we emphasize some aspects that can be used as additional information to improve the search. Vertices of G_2 associated with the same vertex of G_1 should be connected among themselves in real situations, since an over-segmentation method can split an object in several

smaller pieces, but it does not change the piece positions. Regions of the segmented image corresponding to the same region of the model should necessarily be connected. A good strategy to avoid this type of solution is to restrain the search to correspondences for which each set $A(i)$ of vertices induces a connected subgraph in G_2, for every model vertex $i \in N_1$ (connectivity constraint):
Constraint (2): For every $i \in N_1$, the subgraph induced in $G_2(N_2, E_2)$ by $A(i)$ is connected.

Pairs of vertices with null similarity cannot be associated. Such associations are discarded by the constraint below, which strengthens the penalization of associations between vertices with low similarity values induced by the objective function:
Constraint (3): For every $i \in N_1$ and $j \in N_2$, if $s^v(i,j) = 0$, then $x_{ij} = 0$.

Finally, to ensure that all objects of the model appear in the image graph, one additional constraint is imposed:
Constraint (4): For every $i \in N_1$, there exists at least one node $j \in N_2$ such that $(i,j) \in E'$ (i.e., $|A(i)| \geq 1$)).

3 Randomized Construction Algorithm

The construction algorithm proposed in this section is based on progressively associating a node of N_1 with each node of N_2, until a feasible solution x is built. The objective function $f(x)$ does not have to be evaluated from scratch at each iteration. Its value is initialized once for all and progressively updated after each new association between a pair of vertices from N_1 and N_2. Since

$$f^v(x) = \sum_{i \in N_1} \sum_{j \in N_2} (1 - |x_{ij} - s^v(i,j)|) =$$

$$= \sum_{(i,j) \in N_1 \times N_2} (1 - s^v(i,j)) + \sum_{(i,j):x_{ij}=1} (2s^v(i,j) - 1),$$

then $f(x') = f(x) + 2s^v(i,j) - 1$ for any two solutions x and x' that differ only by one additional association between vertices $i \in N_1$ and $j \in N_2$. Similar considerations are used in the evaluation of the term $f^e(x)$, which is increased by $2s^e((a,a'),(b,b')) - 1$ whenever a new pair of edges $(a,a') \in E_1$ and $(b,b') \in E_2$ are associated.

The pseudo-code of the `RandomizedConstruction` randomized algorithm is given in Figure 2. The algorithm takes as parameters the initial seed, the maximum number $MaxTrials$ of attempts to build a feasible solution before stopping, and the maximum number $MaxSolutions$ of solutions built. We denote by $\Gamma_G(j)$ the nodes adjacent to vertex j in a graph G. The number of attempts, the number of solutions built, and the indicator that a feasible solution has been found are initialized in line 1. The optimal value is initialized in line 2. The loop in lines 3–35 performs at most $MaxTrials$ attempts to build at most $MaxSolutions$ solutions. Lines 4–7 prepare and initialize the data for each attempt. The solution x, the set $A(i)$ of nodes associated with each node $i \in N_1$, and the node $A^{-1}(j)$

procedure `RandomizedConstruction`($seed$,$MaxTrials$,$MaxSolutions$)
1. $trials \leftarrow 1$, $solutions \leftarrow 1$, and $feasible \leftarrow$ `.FALSE.`;
2. $f^* \leftarrow -\infty$;
3. **while** $trials \leq MaxTrials$ **and** $solutions \leq MaxSolutions$ **do**
4. $\quad x \leftarrow 0$, $A(i) \leftarrow \emptyset \; \forall i \in N_1$, and $A^{-1}(j) \leftarrow \emptyset \; \forall j \in N_2$;
5. $\quad f^v \leftarrow \sum_{(i,j) \in N_1 \times N_2} (1 - s^v(i,j))$;
6. $\quad f^e \leftarrow \sum_{((i,i'),(j,j')) \in E_1 \times E_2} (1 - s^e((i,i'),(j,j')))$;
7. $\quad V_2 \leftarrow N_2$;
8. $\quad$ **while** $V_2 \neq \emptyset$ **do**
9. $\qquad$ Randomly select a node j from V_2 and update $V_2 \leftarrow V_2 - \{j\}$;
10. $\qquad V_1 \leftarrow N_1$;
11. $\qquad$ **while** $V_1 \neq \emptyset$ **and** $A^{-1}(j) = \emptyset$ **do**
12. $\qquad\quad$ Randomly select a node i from V_1 and update $V_1 \leftarrow V_1 - \{i\}$;
13. $\qquad\quad$ **if** $s^v(i,j) > 0$ **and**
$\qquad\qquad$ the graph induced in G_2 by $A(i) \cup \{j\}$ is connected
14. $\qquad\quad$ **then do**
15. $\qquad\qquad x_{ij} \leftarrow 1$;
16. $\qquad\qquad A(i) \leftarrow A(i) \cup \{j\}$ and $A^{-1}(j) \leftarrow \{i\}$;
17. $\qquad\qquad f^v \leftarrow f^v + 2s_v(i,j) - 1$;
18. $\qquad\qquad$ **forall** $i' \in \Gamma_{G_1}(i)$ **do**
19. $\qquad\qquad\quad$ **forall** $j' \in \Gamma_{G_2}(j)$ **do**
20. $\qquad\qquad\qquad$ **if** $x_{i'j'} = 1$
21. $\qquad\qquad\qquad$ **then** $f^e \leftarrow f^e + 2s^v((i,i'),(j,j')) - 1$;
22. $\qquad\qquad\quad$ **end_forall**;
23. $\qquad\qquad$ **end_forall**;
24. $\qquad\quad$ **end_if**;
25. $\qquad$ **end_while**;
26. $\quad$ **end_while**;
27. $\quad$ **if** $A(i) \neq \emptyset \; \forall i \in N_1$ **and** $A^{-1}(j) \neq \emptyset \; \forall j \in N_2$
28. $\quad$ **then do**
29. $\qquad feasible \leftarrow$ `.TRUE.`;
30. $\qquad solutions \leftarrow solutions + 1$;
31. $\qquad$ Compute $f \leftarrow \alpha/(|N_1| \cdot |N_2|) \cdot f^v + (1-\alpha)/(|E_1| \cdot |E_2|) \cdot f^e$;
32. $\qquad$ **if** $f > f^*$ **then** update $f^* \leftarrow f$ and $x^* \leftarrow x$;
33. $\quad$ **end_if**;
34. $\quad trials \leftarrow trials + 1$;
35. **end_while**;
36. **return** x^*, f^*;

end `RandomizedConstruction`.

Fig. 2. Pseudo-code of the randomized construction algorithm.

associated with each node $j \in N_2$ are initialized in line 4. The terms f^v and f^e are initialized respectively in lines 5 and 6. A temporary copy V_2 of the node set N_2 is created in line 7. The loop in lines 8–26 performs one attempt to create a feasible solution and stops after the associations to each node in V_2 have been investigated. A node $j \in V_2$ is randomly selected and eliminated from V_2 in line

9. A temporary copy V_1 of the node set N_1 is created in line 10. The loop in lines 11–25 searches for a node in N_1 to be associated with node $j \in V_2$. It stops after all possible associations to nodes in N_1 have been investigated without success or if one possible association was found. A node $i \in V_1$ is randomly selected and eliminated from V_1 in line 12. The algorithm checks in line 13 if node i can be associated with node j, i.e., if their similarity is not null and if the graph induced in G_2 by $A(i) \cup \{j\}$ is connected. If this is the case, the current solution and its objective function value are updated in lines 14–24. The current solution is updated in lines 15–16. The term f^v corresponding to the node similarities is updated in line 17. The term f^e corresponding to the edge similarities is updated in lines 18–23. The algorithm checks in line 27 if the solution x built in lines 8–26 is feasible, i.e., if there is at least one node of N_2 associated with every node of N_1 and if there is exactly one node of N_1 associated with every node of N_2. If this is the case, the indicator that a feasible solution was found is reset in line 29 and the number of feasible solutions built is updated in line 30. The value of the objective function for the new solution is computed in line 31. If the new solution is better than the incumbent, then the latter is updated in line 32. The number of attempts to build a feasible solution is updated in line 34 and a new iteration resumes, until the maximum number of attempts is reached. The best solution found x^* and its objective function value f^* are returned in line 36. In case no feasible solution was found, the returned value is $f^* = -\infty$. The complexity of each attempt to build a feasible solution is $O(|N_1|^2 \cdot |N_2|^2)$.

4 Local Search

The solutions generated by a randomized construction algorithm are not necessarily optimal, even with respect to simple neighborhoods. Hence, it is almost always beneficial to apply a local search to attempt to improve each constructed solution. A local search algorithm works in an iterative fashion by successively replacing the current solution by a better solution in the neighborhood of the current solution. It terminates when a local optimum is found, i.e., when no better solution can be found in the neighborhood of the current solution.

We define the neighborhood $N^a(x)$ associated with any solution x as formed by all feasible solutions that can be obtained by the modification of $A^{-1}(j)$ for some $j \in N_2$. For each vertex $j \in N_2$, the candidate set $C(j)$ is formed by all vertices in N_1 that can replace the node currently associated with N_2, i.e. $C(j) = \{k \in N_1 \mid x'$ is a feasible solution, where $x'_{i\ell} = 1$ if $i = k$ and $\ell = j, x'_{i\ell} = 0$ if $i = A^{-1}(j)$ and $\ell = j, x'_{i\ell} = x_{i\ell}$ otherwise$\}$. The number of solutions within this neighborhood is bounded by $|N_1| \cdot |N_2|$.

The pseudo-code of the local search algorithm `LS` using a first-improving strategy based on the neighborhood structure N^a defined above is given in Figure 3. The algorithm takes as inputs the solution x^* built by the randomized construction algorithm and its objective function value f^*. Initializations are performed in lines 1-2. The loop in lines 3-32 performs the local search and stops at the first local optimum of the objective function with respect to the

procedure LS(x^*, f^*)
1. $improvement \leftarrow$ .TRUE.;
2. Build sets $C(j), \forall j \in N_2$;
3. **while** $improvement$ **do**
4. $\quad improvement \leftarrow$ .FALSE.;
5. $\quad$ **forall** $j \in N_2$ **while** .NOT.$improvement$ **do**
6. $\quad\quad i \leftarrow A^{-1}(j)$;
7. $\quad\quad$ **forall** $k \in C(j)$ **while** .NOT.$improvement$ **do**
8. $\quad\quad\quad \Delta^v \leftarrow 2 \cdot s^v(k, j) - 2 \cdot s^v(i, j)$;
9. $\quad\quad\quad \Delta^e \leftarrow 0$;
10. $\quad\quad\quad$ **forall** $j' \in \Gamma_{G_2}(j)$ **do**
11. $\quad\quad\quad\quad$ **forall** $i' \in \Gamma_{G_1}(i)$ **do**
12. $\quad\quad\quad\quad\quad$ **if** $i' = A^{-1}(j')$
13. $\quad\quad\quad\quad\quad$ **then** $\Delta^e \leftarrow \Delta^e + 1 - 2 \cdot s^e((i, i'), (j, j')))$;
14. $\quad\quad\quad\quad$ **end_forall**;
15. $\quad\quad\quad\quad$ **forall** $k' \in \Gamma_{G_1}(k)$ **do**
16. $\quad\quad\quad\quad\quad$ **if** $k' = A^{-1}(j')$
17. $\quad\quad\quad\quad\quad$ **then** $\Delta^e \leftarrow \Delta^e - 1 + 2 \cdot s^e((k, k'), (j, j')))$;
18. $\quad\quad\quad\quad$ **end_forall**;
19. $\quad\quad\quad$ **end_forall**;
20. $\quad\quad\quad \Delta \leftarrow \alpha/(|N_1| \cdot |N_2|) \cdot \Delta^v + (1 - \alpha)/(|E_1| \cdot |E_2|) \cdot \Delta^e$;
21. $\quad\quad\quad$ **if** $\Delta > 0$
22. $\quad\quad\quad$ **then do**
23. $\quad\quad\quad\quad improvement \leftarrow$ .TRUE.;
24. $\quad\quad\quad\quad x^*_{kj} \leftarrow 1, x^*_{ij} \leftarrow 0$;
25. $\quad\quad\quad\quad A(i) \leftarrow A(i) - \{j\}$;
26. $\quad\quad\quad\quad A(k) \leftarrow A(k) \cup \{j\}$;
27. $\quad\quad\quad\quad f^* \leftarrow f^* + \Delta$;
28. $\quad\quad\quad\quad$ Update sets $C(j), \forall j \in N_2$;
29. $\quad\quad\quad$ **end_if**;
30. $\quad\quad$ **end_forall**;
31. $\quad$ **end_forall**;
32. **end_while**;
33. **return** x^*, f^*;

end LS.

Fig. 3. Pseudo-code of the basic local search algorithm using neighborhood N^a.

neighborhood defined by the sets $C(j)$. The control variable is initialized at each local search iteration in line 4. The loop in lines 5-31 considers each node j of graph G_2. The replacement of the node $i = A^{-1}(j)$ currently associated with j (line 6) by each node belonging to the candidate set $C(j)$ is investigated in the loop in lines 7-30. The increase in the value of the objective function due to the node similarity contributions is computed in line 8, while that due to the edge similarity contributions is computed in lines 9-19. If the total increase in the objective function value computed in line 20 is strictly positive (line 21), then the current solution and the control variables are updated in lines 22-28. The procedure returns in line 33 the local optimum found and the corresponding solution value. Each local search iteration within neighborhood N^a has complexity $O(|N_1| \cdot |N_2|^2 + |N_2| \cdot |E_2|)$.

We notice that if $A(i) = \{j\}$ for some $i \in N_1$ and $j \in N_2$ (i.e. $|A(i)| = 1$) then $|C(j)| = \emptyset$, because in this case vertex i would not be associated with any other vertex. It can also be the case that a node $j \in A(i)$ cannot be associated with any other node because $A(i) \setminus \{j\}$ induces a non-connected graph in G_2. In consequence, in these situation the vertex associated with node j cannot be changed by local search within the neighborhood N^a, even if there are other feasible associations. As an attempt to avoid this situation, we define a second neighborhood structure $N^b(x)$ associated with any feasible solution x. This is a swap neighborhood, in which the associations of two vertices $j, j' \in N_2$ are exchanged. A solution $x' \in N^b(x)$ if there are two vertices $i', i'' \in N_1$ and two vertices $j', j'' \in N_2$ such that $x_{i'j'} = 1$, $x_{i''j''} = 1$, $x'_{i'j''} = 1$, and $x'_{i''j'} = 1$, with all other associations in solutions x and x' being the same.

Local search within the swap neighborhood N^b has a higher time complexity $O(|N_2|^2 \cdot |E_2|)$ than within neighborhood N^a. Also, $|N^b(x)| >> |N^a(x)|$ for any feasible solution x. Accordingly, we propose an extended local search procedure `LS+` which makes use of both neighborhoods. Whenever the basic local search procedure `LS` identifies a local optimum x^* with respect to neighborhood N^a, the extended procedure starts a local search from the current solution x^* within neighborhood N^b. If this solution is also optimal with respect to neighborhood N^b, then the extended procedure stops; otherwise algorithm `LS` resumes from any improving neighbor of x^* within N^b.

5 Computational Results

The algorithms described in the previous sections were implemented in C and compiled with version 2.96 of the `gcc` compiler. We used an implementation in C of the random number generator described in [18]. All computational experiments were performed on a 450 MHz Pentium II computer with 128 Mbytes of RAM memory, running under version 7.1 of the Red Hat Linux operating system.

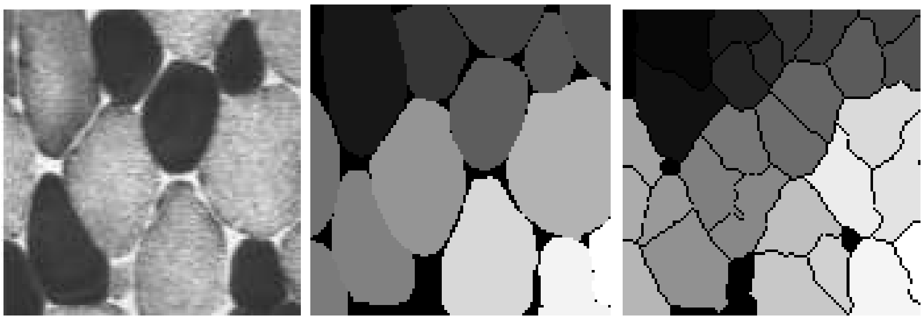
Unlike other problems in the literature, there are no benchmark instances available for the problem studied in this paper. We describe below a subset of seven test instances used in the evaluation of the model and the algorithms proposed in Sections 3 and 4.

Instances GM-5, GM-8, and GM-9 were randomly generated [1], with node and edge similarity values in the interval [0,1]. Instance GM-8 was also used in the computational experiments reported in [1]. Instances GM-5 and GM-8 have isolated nodes: two in the image graph G_2 of GM-5 and three in the model graph G_1 of GM-8. Instances GM-5a and GM-8a are derived from them, by the introduction of additional edges to connect the isolated nodes.

Instances GM-6 and GM-7 were provided by Perchant and Bengoetxea [12, 14] and built from real images. Instance GM-6 was built from magnetic resonance images of a human brain, as depicted in Figure 4. Instance GM-7 was created for the computational experiments reported in [14] from the 2D images given in Figure 5. The image (a) was over-segmented in 28 regions (c) and compared with a model with only 14 well defined regions (b). The model graph G_1 has 14 vertices and 27 edges, while the over-segmented image graph G_2 has 28 vertices and 63 edges. Grey levels were used in the computation of node similarities, while distances and adjacencies were used for the computation of edge similarities.

(a) (b) (c)

Fig. 4. Instance GM-6: (a) original image, (b) model, and (c) over-segmented image.



(a) (b) (c)

Fig. 5. Cut of a muscle (instance GM-7): (a) original 2D image, (b) model, and (c) over-segmented image.

We summarize the characteristics of instances GM-5 to GM-9 in Table 1. For each instance, we first give the number of nodes and edges of the model and image graphs. We also give the optimal value f^* obtained by the exact integer programming formulation proposed by Duarte [8] using the mixed integer programming solver CPLEX 9.0 and the associated computation time in seconds on a 2.0 GHz Pentium IV computer (whenever available), considering exclusively the vertex contribution to the objective function. In the last two columns, we give the value f^{LS+} of the solution obtained by the randomized construction algorithm followed by the application of the extended local search procedure LS+ and the total computation time in seconds, with the maximum number of attemps to find a feasible solution set at $MaxTrials = 500$ and the maximum number of feasible solutions built set at $MaxSolutions = 100$.

Table 1. Characteristics and exact results for instances GM-5 to GM-9.

Instance	$\|N_1\|$	$\|E_1\|$	$\|N_2\|$	$\|E_2\|$	f^*	time (s)	f^{LS+}	time (s)
GM-5	10	15	30	39	0.5676	7113.34	0.5534	0.01
GM-5a	10	15	30	41	0.5690	2559.45	0.5460	0.02
GM-6	12	42	95	1434	0.4294	23668.17	0.4286	2.68
GM-7	14	27	28	63	0.6999	113.84	0.6949	$< 10^{-3}$
GM-8	30	39	100	297	(a) 0.5331	(a) 4.27	0.5209	1.02
GM-8a	30	42	100	297	(a) 0.5331	(a) 4.12	0.5209	1.02
GM-9	50	88	250	1681	–	–	0.5204	42.26

(a) linear programming relaxation
–: not available

The results in Table 1 illustrate the effectiveness of the heuristics proposed in this work. The non-bijective graph matching problem can be exactly solved only for small problems by a state-of-the-art solver such as CPLEX 9.0. Even the medium size instances GM-8 and GM-8a cannot be exactly solved. Only the linear programming bounds can be computed in resonable computation times for both of them. On the other hand, the combination of the randomized construction algorithm with the local search procedure provides high quality solutions in very small computation times. Good approximate solutions for the medium size instances GM-8 and GM-8a (which were not exactly solved by CPLEX) within 2.3% of optimality can be easily computed in processing times as low as one second.

Table 2 illustrates the results obtained by the randomized construction algorithm and the extended local search procedure for instances GM-5 to GM-9 with $\alpha = 0.9$. The maximum number of attemps to find a feasible solution was fixed at $MaxTrials = 500$ and the maximum number of feasible solutions built was fixed at $MaxSolutions = 100$. For each instance, we give the number of attempts necessary to find the first feasible solution, the value $f^{(1)}$ of the first feasible solution found, the number of attempts necessary to find the best among the first 100 feasible solutions built, the value $f^{(100)}$ of the best feasible solution found, and the average computation time per attempt in seconds. The last three columns report statistics for the extended local search algorithm: the number of local search iterations until local optimality, the value f^{LS+} of the best solution found, and the average computation time per iteration in seconds.

Table 2. Results obtained by the randomized construction algorithm and the extended local search procedure with $MaxTrials = 500$ and $MaxSolutions = 100$.

Instance	first	$f^{(1)}$	best	$f^{(100)}$	time (s)	iteration	f^{LS+}	time (s)
GM-5	297	0.5168	297	0.5168	$< 10^{-3}$	19	0.5474	0.002
GM-5a	9	0.4981	417	0.5243	$< 10^{-3}$	13	0.5434	0.002
GM-6	1	0.4122	40	0.4168	0.001	320	0.4248	0.020
GM-7	5	0.6182	34	0.6282	$< 10^{-3}$	12	0.6319	0.001
GM-8	26	0.4978	292	0.5022	0.002	118	0.5186	0.014
GM-8a	26	0.5014	292	0.5058	0.002	120	0.5222	0.014
GM-9	1	0.5049	207	0.5060	0.010	511	0.5187	0.134

The computation time taken by each attempt of the randomized construction algorithm to build a feasible solution is very small, even for the largest instances. The algorithm is very fast and finds the first feasible solution in only a few attempts, except in the cases of the difficult instances with isolated nodes. However, even in the case of the hard instance GM-5, the algorithm managed to find a feasible solution after 297 attempts. For the other instances, the construction algorithm found a feasible solution in very few iterations. Even better solutions can be obtained if additional attempts are performed.

The local search algorithm improved the solutions built by the construction algorithm for all test instances. The average improvement with respect to the value of the solution obtained by the construction algorithm was approximately 3%.

6 Concluding Remarks

We formulated the problem of finding the best correspondence between two graphs representing a model and an over-segmented image as a combinatorial optimization problem.

A robust randomized construction algorithm was proposed to build feasible solutions for the graph matching problem. We also proposed a local search algorithm based on two neighborhood structures to improve the solutions built by the construction algorithm. Computational results were presented to different test problems. Both algorithms are fast and easily found feasible solutions to realistic problems with up to 250 nodes and 1681 edges in the graph representing the over-segmented image. The local search algorithm consistently improved the solutions found by the construction heuristic. Both algorithms can be easily adapted to handle more complex objective function formulations.

Besides the quality of the solutions found, the randomized algorithm may also be used as a robust generator of initial solutions for population methods such as the genetic algorithm described in [14], replacing the low quality randomly generated solutions originally proposed. The construction and local search algorithms can also be put together into an implementation of the GRASP metaheuristic [16].

References

1. E. Bengoetxea, P. Larranaga, I. Bloch, A. Perchant, and C. Boeres. Inexact graph matching by means of estimation distribution algorithms. *Pattern Recognition*, 35:2867-2880, 2002.
2. I. Bloch. Fuzzy relative position between objects in image processing: a morphological approach. *IEEE Transactions on Pattern Analysis Machine Intelligence*, 21:657–664, 1999.
3. I. Bloch. On fuzzy distances and their use in image processing under imprecision. *Pattern Recognition*, 32:1873–1895, 1999.
4. I. Bloch, H. Maître, and M. Anvari. Fuzzy adjacency between image objects. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 5:615–653, 1997.
5. K.P. Chan and Y.S. Cheung. Fuzzy-attribute graph with application to chinese character recognition. *IEEE Transactions on Systems, Man and Cybernetics*, 22:402–410, 1992.
6. A.D.J. Cross and E.R. Hancock. Relational matching with stochastic optimization. In *International Conference on Computer Vision*, pages 365–370, 1995.
7. A.D.J. Cross, R.C. Wilson, and E.R. Hancock. Inexact graph matching using genetic search. *Pattern Recognition*, 30:953–970, 1997.
8. A.R. Duarte. *New heuristics and an exact integer programming formulation for an inexact graph matching problem* (in Portuguese). M.Sc. Dissertation, Catholic University of Rio de Janeiro, 2004.
9. Y. El-Sonbaty and M. A. Ismail. A new algorithm for subgraph optimal isomorphism. *Pattern Recognition*, 31:205–218, 1998.
10. A. W. Finch, R. C. Wilson, and E. R. Hancock. Symbolic Graph matching with the EM algorithm. *Pattern Recognition*, 31:1777–1790, 1998.
11. H. Moissinac, H. Maître, and I. Bloch. Markov random fields and graphs for uncertainty management and symbolic data fusion in a urban scene interpretation. *EUROPTO Conference on Image and Signal Processing for Remote Sensing*, 2579:298–309, 1995.
12. A. Perchant. *Morphisme de graphes d'attributs flous pour la reconnaissance structurelle de scènes*. Doctorate thesis, École Nationale Supérieure des Télécommunications, 2000.
13. A. Perchant and I. Bloch. A new definition for fuzzy attributed graph homomorphism with application to structural shape recognition in brain imaging. In *Proceedings of the 16th IEEE Conference on Instrumentation and Measurement Technology*, pages 402–410, 1999.
14. A. Perchant, C. Boeres, I. Bloch, M. Roux, and C.C. Ribeiro. Model-based scene recognition using graph fuzzy homomorphism solved by genetic algorithm. In *2nd IAPR-TC-15 Workshop on Graph-based Representations*, pages 61–70, 1999.

15. H.S. Ranganath and L.J. Chipman. Fuzzy relaxaton approach for inexact scene matching. *Image and Vision Computing*, 10:631–640, 1992.
16. M.G.C. Resende and C.C. Ribeiro. "Greedy randomized adaptive search procedures". *Handbook of Metaheuristics* (F. Glover and G. Kochenberger, eds.), pages 219-249, Kluwer, 2002.
17. A. Rosenfeld, R. Hummel, and S. Zucker. Scene labeling by relaxation operations. *IEEE Transactions on Systems, Man and Cybernetics*, 6:420–433, 1976.
18. L. Schrage. A more portable FORTRAN random number generator. *ACM Transactions on Mathematical Software*, 5:132-138, 1979.
19. M. Singh and A. C. S. Chaudhury. Matching structural shape descriptions using genetic algorithms. *Pattern Recognition*, 30:1451–1462, 1997.
20. A.K.C. Wong, M. You, and S.C. Chan. An algorithm for graph optimal monomorphism. *IEEE Transactions on Systems, Man and Cybernetics*, 20:628–636, 1990.
21. E.K. Wong. Model matching in robot vision by subgraph isomorphism. *Pattern Recognition*, 25:287–303, 1992.