Chapter 3

Graph matching as a combinatorial optimization problem with constraints

'The beginning of knowledge is the discovery of something we do not understand.'

Frank Herbert

3.1 Introduction

Let us introduce some notations. We will call $G_M = (V_M, E_M)$ the model graph and $G_D = (V_D, E_D)$ the data graph which contains the image segments that have to be matched against G_M, where V_i is the set of vertices and E_i is the set of edges of graph G_i ($i = M, D$). In inexact graph matching problems G_D is assumed to contain more segments than G_M, as that is the case when generated from an over-segmented image. Usual constraints for the matching are that each vertex of G_D is matched with exactly one vertex of G_M (which assumes that no unexpected objects are present on the image data) and that each vertex of G_M is matched at least with one vertex of G_D (which assumes that all model objects are indeed present in the image)[1]. In Section 3.2, we summarize how such constraints are taken into account in existing methods.

In order to solve any problem using combinatorial optimization techniques, it is necessary to find a means to represent each of the solutions to the problem as a vector of values (usually this is known as an *individual*), as well as a *fitness function* to evaluate each of these solutions so that the algorithm can distinguish among good and bad solutions. In Section 3.3, we address the problem of representing solutions as individuals. Constraints are introduced either in the representation itself, or as an elimination process of incorrect individuals. In Section 3.4 we deal with the definition of fitness functions.

[1]This implies in particular that the method is not directly applicable to incomplete models or to images with pathologies (in the case of medical images).

3.2 Graph matching problems with special constraints in the literature

Some real graph matching problems contain more specific constraints that any valid homomorphism has to satisfy apart from the ones commented so far. These constraints are very specific for each graph matching problem, but they usually taken into account when generating the graph attributes. [Feder and Vardi, 1999] propose a framework based on group theory for constraint satisfaction in NP-complete problems such as attributed graph isomorphism. Authors claim that this framework can also be applied to any other type of graph matching problems. Also, the complexity of rules and constraints in conceptual graphs is analyzed in [Baget and Mugnier, 2002]. In [Blake, 1994] the idea of partitioning the graph matching problem into sub-problems under the control of constraints is considered. Additionally, in [Liu, 1997b] a method that provides a design advisory system for mechanical components is developed by building a scheme for modelling constraints by defining a new type of graphs called *constraint graphs*. In this work, the representation and management of constraints are elaborated using a graph-based approach in the form of constraint graphs. The product design optimization is formulated as a graph matching problem and solved by integer programming techniques.

However, regarding graph matching algorithms and methods to find correct solutions, one of most applied mechanisms for obtaining only valid solutions at the end of the search in problems with constraints is the use of a particular type of neural networks called Hopfield networks [Suganthan et al., 1995, Suganthan and Yan, 1998, Suganthan et al., 1999]. This approach encodes the constraints of the problem in a way that a fitness function does not have to take them into account. This is achieved due to the ability of the network to learn the constraint parameter adaptively, as the adaptation scheme eliminates the need to specify the constraint parameter empirically and generates valid and better quality mappings. Another approach is proposed in [Gangnet and Rosenberg, 1993], which is an illustration on how to apply constraints using the method proposed for constraint solving in [Freeman-Benson et al., 1990]. The authors claim that their approach is simple and efficient for constraint resolution when applied to graph matching problems. Finally, a mean field annealing neural network is proposed as a constraint satisfaction network in [Lyul and Hong, 2002].

Examples of real graph matching problems with constraints introduced in the literature are frontal face authentication [Ma and Xiaoou, 2001, Tefas et al., 2001], and topographical constraints in face recognition [Wiskott, 1999].

3.3 Representations of graph matching solutions by means of individuals

A solution to a graph matching problem is an association between vertices of G_M and vertices of G_D satisfying the required constraints of the particular problem. Regarding the way of representing a solution by means of individuals, the type of values that these can contain allow us to classify the different individual formats or representations as

- discrete individuals: all the values in the individual are discrete,
- continuous individuals: all the values in the individual are continuous.

The type of individual representation used is also a factor that is determined by the type of combinatorial optimization technique to be applied[2]. For instance, Genetic Algorithms (GAs) are usually applied only to discrete individuals, while other techniques such as Evolutionary Strategies (ESs) can only be applied to continuous individuals. EDAs can be applied to both types of individuals, as under this paradigm there are many different algorithms available that will be described in the next chapter.

It is important to note that, independently of using discrete or continuous individuals, different individual representations can be used for a same problem. When selecting an individual representation it should be taken into account that this choice is an important factor that will condition the evolution of the search process. Just as an example on the type of factor that should be considered, it is convenient not to use ambiguous individual representations in which two different individuals represent a same solution for the problem. However, in some cases the individual representation used allows the existence of ambiguities but this aspect is controlled deliberately.

3.3.1 Individual representations in the discrete domain

We denote by $V_M = \{u_M^1, u_M^2, \ldots, u_M^{|V_M|}\}$ and $V_D = \{u_D^1, u_D^2, \ldots, u_D^{|V_D|}\}$ the set of vertices of G_M and G_D respectively. There are different ways of representing individuals with discrete values for inexact graph matching, two examples of which are as follows:

Representation 1: Individuals with $|V_M| \cdot |V_D|$ genes or variables c_{ij}, that only take values 0 and 1.

For $1 \leq i \leq |V_M|$ and $1 \leq j \leq |V_D|$, $c_{ij} = 1$ means that the vertex u_D^j of G_D is matched with the vertex u_M^i of G_M.
This is the representation used for instance in [Boeres et al., 1999].

Representation 2: Individuals with $|V_D|$ genes or variables, X_i $i = 1, \ldots, |V_D|$, where each of them that can take any value between 1 and $|V_M|$.

For $1 \leq k \leq |V_M|$ and $1 \leq i \leq |V_D|$, $X_i = k$ means that the vertex u_D^i of G_D is matched with the u_M^k vertex of G_M.

The latter is the representation of individuals that we have selected for EDAs and GAs for reasons that will be explained in Section 3.3.3. Therefore, in this case individuals contain as many variables as vertices are in G_D ($|V_D|$ vertices), and each of the variables can take as many values as vertices are in G_M ($|V_M|$ values).

The biggest drawback of using any of these two representations is that some individuals can represent solutions that are not acceptable for the problem, that is, that do not satisfy a set of constraints defined beforehand. In this thesis we will deal with very particular constraints. These will be discussed later in Section 3.3.3.

Representation 3: Individual representation based on a permutation of values. The individual can also be represented as a permutation of discrete values. A permutation is a list of numbers in which all the values from 1 to n have to appear in an individual of size n. In this case, the values of the individual do not represent directly the matching of each

[2]It must be said that even in the case of using a continuous individual representation we will still have a combinatorial optimization problem since in our approach continuous individuals are transformed to a permutation of discrete values. This procedure is explained later in this chapter.

G_D, but the order in which each vertex of G_D will be matched following a predefined procedure. Permutation-based individual representations have been typically applied to problems such as the Travelling Salesman Problem [Flood, 1956] or the Vehicle Routing Problem [Fiala, 1978]. An illustrative example of applying permutations to genetic algorithms for solving these problems can be found in [Freisleben and Merz, 1996]

For the particular graph matching problems in this thesis, we have applied individual representations 2 and 3 for the discrete domain. In both cases, discrete individuals have a length of $|V_D|$ variables, where the number of values that each variable can take are $|V_M|$ or $|V_D|$ for Representations 2 and 3 respectively. In addition, due to the type of graph matching problems that this thesis deals with, we have decided to add constraints that have to be satisfied by any valid solution (independently of the representation used), which are defined in Section 3.3.3

From the permutation to the solution it represents

Once having the permutation, the individual has to be translated to the solution it symbolizes so that it can be evaluated. Because the evaluation of an individual is executed many times by any graph matching algorithm, it is important that this translation is performed by a fast and simple algorithm.

Evaluating a solution requires to compare vertices of G_M and vertices of G_D. If we have a permutation, a solution can be evaluated by comparing the vertices in the order given by the permutation and deciding which is the most similar by means of a similarity function, $\varpi(i, j)$, defined to compute the similarity between vertices of G_D. The similarity measures used so far in the literature have been applied to two vertices [Perchant et al., 1999, Perchant and Bloch, 1999, 2000b, Perchant, 2000, Perchant and Bloch, 2000a], one from each graph, and their goal has been to help in the computation of the fitness of a solution, that is, the final value of a fitness function.

Figure 3.1 shows a procedure that could be used in order to translate a permutation of discrete values –an individual $\boldsymbol{x}= (x_1, \ldots, x_{|V_M|}, x_{|V_M|+1}, \ldots, x_{|V_D|})$– to the solution that it represents. This procedure follows an idea inspired on the partitional clustering algorithms proposed in [McQueen, 1967] and specially in [Forgy, 1965], and it is divided in two main steps as follows:

- In the first step the values x_1 to $x_{|V_M|}$ are directly matched to vertices of G_M from 1 to $|V_M|$ respectively.
- For each of the next values of the individual, $x_{|V_M|+1}$ to $x_{|V_D|}$, and again following the order given in the permutation, the most similar vertex of G_D based on the similarity measure $\varpi(i, j)$ will be selected, and its previously matched vertex of G_M will be also chosen as the matching for the new vertex.

Permutation-based representations can be used for any graph matching problem. A detailed example of this method, as well as a deeper explanation of it can be found in Appendix A and in [Bengoetxea et al., 2001c,d].

Another important aspect of using a permutation-based approach is the fact that the cardinality of the search space is $|V_D|!$, which is different from the $|G_M|^{|G_D|}$ cardinality of

```
From permutations to the solution
Definitions
    $|V_M|$: number of vertices in the model graph $G_M$
    $|V_D|$: number of vertices in the data graph $G_D$ (where $|V_D| > |V_M|$)
    Size of the individual (the permutation): $|V_D|$
    $\boldsymbol{x}= (x_1, \ldots, x_{|V_D|})$: individual containing a permutation
    $x_i \in \{1, \ldots, |V_D|\}$: value of the $i^{th}$ variable in the individual
    $PV_i = \{x_1, \ldots, x_{i-1}\}$: set of values assigned in the individual to
        the variables $X_1, \ldots, X_{i-1}$. ($PV$ = previous values)
    $\varpi(i, j)$: similarity function that measures the similarity between
        vertex $i$ and vertex $j$ (with $i, j \in V_D$)
Procedure
    Phase 1
        For $i = 1, 2, \ldots, |V_M|$
        (first $|V_M|$ values in the individual, treated in order)
            Match vertex $x_i \in V_D$ of data graph $G_D$
                with vertex $i \in V_M$ in model graph $G_M$

    Phase 2
        For $i = |V_M| + 1, \ldots, |V_D|$
        (remaining values in the individual, treated in this order)
            Let $k \in PV_i$ be the most similar vertex to $x_i$ from
                all the vertices of $PV_i$ ($k = \arg\max_{j=1 \ldots i-1} \varpi(i, j)$)
            Match vertex $x_i \in V_D$ of data graph $G_D$
                with the vertex that is matched to vertex $k$ of $G_M$
```

Figure 3.1: Pseudocode to compute the solution represented by a permutation-based individual.

the previously described individual representation. Moreover, the fact of using permutations and a similarity measure $\varpi(i, j)$ leads to redundancies in the solutions, as two different permutations may correspond to the same solution. An example of this is also shown again in Appendix A and in [Bengoetxea et al., 2001a,c].

Definition of the similarity

The definition of the similarity function $\varpi(i, j)$ is very important in the translation procedure from a permutation-based individual to the solution it symbolizes. There are three main aspects to be taken into account when defining this function for its use in the second step of the translation procedure:

1. **Which vertices have to be compared.** The two vertices to compare can be of graph G_D, or both from graph G_D (e.g. $\varpi(i, j)$ has two parameters, vertex i and vertex j, we know that $i \in V_D$, the choice is either $j \in V_M$ or $j \in V_D$). Other approaches are also possible, for instance combining the similarity of vertices from both G_M and G_D

and assigning them a weight, or also by having a fitness function capable of returning a value for individuals that are not correct permutations.

2. **Recalculating or not the similarity measure as the individual is being generated.** The function $\varpi(i,j)$ could be constant or not for any two vertex values. In the latter case, the similarity value can be assumed to vary depending on the vertices that have already been matched while the individual is being instantiated. In that case, each time that a variable is instantiated, the values of $\varpi(i,j)$ will vary depending on the effect of the new and the previous matchings. In other words, this means that in the second step of the translation an extra clustering procedure would be required, such as the cluster analysis proposed in [Forgy, 1965], in order to update the function $\varpi(i,j)$.

3. **Selection of the attributes of both vertices and edges that will be used for measuring the similarity.** This aspect is specific for each particular problem. This choice will determine to an important degree the behavior of the algorithm.

In [Bengoetxea et al., 2001c,d] we propose a similarity measure $\varpi(i,j)$ that is used to measure the similarity between vertices of the same graph G_D, which has fixed similarity values and therefore no variations apply during the instantiation of individuals.

3.3.2 Individual representations in the continuous domain

Continuous EDAs provide different algorithms for the continuous domain that could be more suitable for some problems (see Section 4.4). Nevertheless, the representation of individuals has to be defined in the most appropriated way to obtain the best performance with this approach.

Individuals in this approach consist of a continuous value in $\Re^n$. The main goal is to find a representation of individuals and a procedure to obtain an univocal solution to the matching from each of the possible permutations.

For this case we propose as an example a strategy based on the mechanism of the previous section, trying to translate the individual in the continuous domain to a correct permutation in the discrete domain. This translation will give us a permutation of discrete values, and we would proceed as explained in Section 3.3.1 in order to obtain the solution that the individual symbolizes.

Again, this new representation of individuals does not give a direct meaning of the solution it represents. This new type of representation can also be regarded as a way to change the search from the discrete to the continuous world, which allows us to apply techniques to estimate densities that are completely different from the ones used in discrete domains.

As this procedure to translate from the continuous world to the discrete one has to be performed for each individual as an additional step to the method introduced in the previous section, this process has to be fast and simple enough in order to reduce computation time.

Taking these aspects into account, we show as an example a method that can easily be applied to graph matching problems. The individual size is set to $|V_D|$ as before, where each of the variables of the individual can take any value following a Gaussian distribution. In order to obtain a translation to a discrete permutation, we propose to order the continuous values of the individual, and to set its corresponding discrete values by assigning to each $x_i \in \{1, \ldots, |V_D|\}$ the respective order in the continuous individual. The procedure described in this section is shown as pseudocode in Figure 3.2.

```
From a continuous value in $\Re^n$ to a permutation
Definitions
    $n = |V_D|$: size of the individual, which is the number of
        vertices in data graph $G_D$ (the permutation)
    $\boldsymbol{x^C} = (x_1^C, \ldots, x_{|V_D|}^C)$: individual containing continuous
        values (the input)
    $\boldsymbol{x^D} = (x_1^D, \ldots, x_{|V_D|}^D)$: individual containing a permutation
        of discrete values (the output)
    $x_i^D \in 1, \ldots, n$: value of the $i^{th}$ variable in the individual
Procedure
        Order the values $x_1^C, \ldots, x_{|V_D|}^C$ of individual $\boldsymbol{x^C}$ using any
            fast sorting algorithm such as Quicksort
        Let $k_i$ be position in which each value $x_i^C$ $(1 \leqslant i \leqslant |V_D|)$ occupies
            after ordering all the values
        The values of the individual $\boldsymbol{x^D}$ will be set in the following way:
            $\forall i = 1, \ldots, |V_D| \;\; x_i^D = k_i$
```

Figure 3.2: Pseudocode to translate from a continuous value in $\Re^n$ to a discrete permutation composed of discrete values.

3.3.3 Conditions for a correct individual in a graph matching problem

Each of the different image recognition problems that are solved using graph matching techniques has particularities that have to be taken into account by any acceptable solution. Due to this, the way of considering the constraints in a graph matching approach is an important aspect that has to be considered attentively. A useful way of dealing implicitly with constraints is to select an individual representation that does not allow the possibility for invalid individuals to appear. Unfortunately, this representation does not always exist, and satisfaction of the constraints need to be tackled using explicit mechanisms.

In order to show how problem specific constraints can be tackled explicitly, independently of the representation of individuals of choice, we will consider that any individual to represent a *correct solution* for graph matching examples in this thesis satisfies all the following 3 conditions:

1. All the vertices in G_D must have a corresponding match with a vertex in G_M. For some type of images such as the ones in cartography this condition is not necessary: sometimes when comparing a new photograph with a previous map of the same area additional objects such as new roads and new houses that cannot be matched could appear. For these cases, the use of a dummy vertex is advised (this technique is described in Section 2.2.2). In our graph matching problems in this thesis we do not consider such cases and assume that all of our segments correspond at least to a vertex in G_M, and as a result no dummy vertex has to be defined (i.e. $\emptyset$ label has to be added).

2. Each vertex in G_D can have at most a vertex matched in G_M. It is not acceptable that a segment in the image corresponds to more than a segment in the atlas. This aspect is analyzed in general terms for graph matching problems in Section 2.2.3. This condition is satisfied in the human brain structure recognition problem [Perchant and Bloch, 1999] by applying over-segmentation techniques to the image, making sure that an object in the model appears always properly divided in the segmented image. None of the individual representations defined described in Section 3.3.1 do not allow matching a vertex of G_D against two or more segments in G_M, and thus using any of those representations this condition would not need to be controlled in the graph matching algorithms (Section 4.2.2 explains more details about controlling the generation of individuals).

3. All the vertices in G_M must have at least a matched vertex of G_D. We have decided to add this additional assumption to the graph matching problems in this thesis, because it also needs to be satisfied in some real graph problems such as the one that is introduced in Section 7.2 for the recognition of human brain images.

It is important to ensure that the final solution of any graph matching algorithm corresponds to a correct individual. Moreover, whichever the representation of individuals chosen, any acceptable individual obtained with any of the methods proposed in GAs, EDAs, or any other graph matching approach in order to be considered as acceptable for our problem.

The reason for considering constraints only for vertices and not for the edges is that when using graph matching for image recognition in vertices represent image regions, and the only constraints that considered here are about the properties that the best solution must satisfy. In this sense, the arcs are used for representing information about relations between the regions, but the final solution simply shows the matching vertex by vertex. Constraints such as taking into account only some specific edges are also considered in these problems, but these are applied when defining the fitness functions. The main difference between many fitness functions proposed in Section 3.4 are actually the edges that are considered or ignored.

The choice of a particular representation of individuals reviewed in Section 3.3.1 is important to reduce the effort of controlling the satisfaction or not of these three conditions. If we compare Representations 1 and 2 all the requirements to check whether an individual is correct or not can be summarized as follows (the need to analyze the first condition is not required since we have decided not to use dummy vertices in our problems):

1. Every vertex of G_D must be matched with one and only one vertex of G_M (2 first conditions).

- **Representation 1:**

$$\sum_{i=1}^{|V_M|} c_{ij} = 1 \quad \forall j \in \ \{1, \ldots, |V_D|\}$$

- **Representation 2:**

  This condition is inherent in the representation, there is no need to check this condition.

2. Every vertex of G_M must be matched at least with one vertex of G_D (third condition).

1	1	1	2	2	2	2	2	3	4	5	6	7	8

(a)

1	1	1	1	2	2	2	2	3	3	3	3	4	4

(b)

Figure 3.3: Representation of a correct individual (a), and an incorrect individual (b) for our graph matching problem example, for the case that G_M (the model graph) contains 8 vertices and G_D (the data graph representing the segmented image) contains 14 vertices.

- **Representation 1:**

$$\forall i \in \ \{1, \ldots, |V_M|\} \ \ \exists j \ \ c_{ij} = 1$$

- **Representation 2:**

$$\forall i \in \ \{1, \ldots, |V_M|\} \ \ \exists j \ \ X_j = i$$

This shows that the Representation 2 of individuals only requires to check actively the third constraint. This is the reason why this representation will be used in our experiments on this thesis. Note that the constraints we impose are a simplification of more general inexact graph matching problems. The interest is that they restrict the set of possible solutions.

The case of Representation 3 of individuals of Section 3.3.1 is not comparable with the other two ones. The reason for this is that the individuals themselves are not the ones that have to satisfy these three conditions, and it is the solution that they symbolize the one that needs to satisfy them. In fact, following the procedure described in Section 3.3.1 we ensure that any individual will satisfy these constraints. As a result, we do not need to worry about the correctness of the particular solutions when using this third representation of individuals. However, the single constraint that these individuals must satisfy is that they must contain a correct permutation, which becomes in practice a more restrictive property to satisfy than the three conditions introduced in this section.

In order to illustrate the meaning of these constraints, we will consider as an example a model graph G_M containing 8 vertices (labelled from 1 to 8) and a data graph G_D representing the segmented image which contains 14 vertices (labelled from 1 to 14). If we use the Representation 2 of individuals of Section 3.3.1 for our graph matching problem, the individual in Figure 3.3a shows an example of a correct matching, where the first three vertices of G_D are matched with the vertex number 1 of G_M, the next five vertices of G_D are matched with the vertex number 2 of the graph G_M, and so on. In such a representation, when generating new individuals the result can be an invalid solution (i.e. it does not satisfy our constraint). The individual in Figure 3.3b is an example of an invalid solution for our problem. In this case, the individual represents a matching for some of the vertices of G_M, but there are still other vertices from this model graph (vertices 5, 6, 7 and 8) that have no match at all.

3.3.4 What to do with incorrect individuals?

It is important to decide what to do with these incorrect individuals. In the literature many papers can be found in the domain of GAs where the existence of incorrect individuals is allowed hoping that these individuals can lead to the generation of fitter correct individuals. In many other articles [Richardson et al., 1989, Smith and Tate, 1993], individuals not representing a valid solution for the problem are either corrected or penalized. Here, we consider all these possibilities and compare them to each other (see Section 4.5). EDAs have been applied to problems with constraints in very few cases [Bengoetxea et al., 2000, 2001a,b], and this dissertation shows and introduces many ways on which EDA approaches can be adapted for this type of problems taking as an example a combinatorial optimization problem with constraints such as inexact graph matching.

3.4 Fitness functions applied to graph matching

The objective of the fitness function is to have a means to evaluate each of the possible individuals so that the search algorithm can compare the different solutions and act in consequence to find the best solution. Therefore, it is important to define appropriately this function in order to assess the search of any graph matching algorithm. The behavior of the fitness function is also very dependent on the individual representation selected and both elements are very linked to each other. The influence of the fitness function definition in finding the best matching between graphs has also been subject of research [Bunke, 1999].

This section discusses the different possibilities for defining a fitness function. The decision of which fitness function to use is very important when using metaheuristics, as important as the individual representation that is used, because it will also determine how difficult is the search for an optimal individual that will lead to a correct recognition.

An ideal fitness function should verify the following properties (assuming that the best fitness is the highest):

1. It should be monotonic, that is, the closer an individual is to the optimum solution the larger its fitness has to be. This assumption is very strong since local maxima are very often present, but these should be avoided as much as possible. However, above all, the most important is that the optimum solution has the largest value.

2. It has to avoid ambiguities, and therefore different individuals should have a different fitness value when they are evaluated.

3. The fitness function has to take into account only the appropriate vertex and edge attributes, the ones that are more meaningful in the search for the optimum solution. In many cases, it is also important to assign a different weight to vertex and edge similarities, giving more importance to the edge attributes as these are the ones that take best into account relationships between regions and not yet the regions themselves.

4. As a secondary aim, it has to be easy and fast to compute, as this is executed many times and it could otherwise delay too much the execution of the whole algorithm.

Whichever the fitness function selected, it is important to take into account that the fitness function is always very particular for each problem, and that the fitness function itself influences the behavior and the final results obtained by the graph matching algorithm. In addition, main differences appear between fitness functions defined for discrete and continuous domains. In the next sections fitness functions for discrete representations are analyzed, while in the last one we focus on fitness functions for the continuous domain.

3.4.1 Graph attributes and similarities

Both the model and data graphs are attributed, and the recognition process is based on similarities between these attributes in the form of fitness functions. The type of attributes that can be used for the computation of the fitness function and the similarity between graphs can be based on different paradigms. Examples of this are the use of fuzzy set theory following the theoretical background developed in [Bloch, 1999a,b]. For instance, many references on fuzzy set theory applied to graph matching concentrate on building dedicated fitness functions. In [Perchant et al., 1999, Perchant and Bloch, 1999] the fitness function is based on the fuzzy attributes of each of the model and data graphs and the fitness value of each solution is computed from comparing directly the similarity between these, whereas in [Suganthan et al., 1998] the fitness function is computed by fusing the information encoded in the attributes using nonlinear fuzzy information aggregation operators. In both cases the learning of the different matching for each of the attribute-pairs is formulated as an optimization problem, although in the former a genetic algorithm is used and in the latter a learning procedure based on gradient projection algorithm is applied.

Apart from using fuzzy set theory for graph matching, other alternatives are the use of vector-type and geometric attributes, and attributes based on probability theory using distribution divergences and likelihood values. We will provide examples of these attribute types.

Once having defined vertex and edge attributes, similarities between attributes of the model and of the data are defined. The similarity between any two vertices $a_D \in V_D$ and $a_M \in V_M$, denoted by $c_N(a_D, a_M)$, is defined in terms of vertex attributes and their semantics, and we assume to be normalized by returning a value in $[0, 1]$ where a higher value represents a higher similarity. Analogously, the similarity between two edges $e_D = (a_D^i, a_D^j)$ of E_D and $e_M = (a_M^k, a_M^l)$ of E_M is denoted by $c_E(e_D, e_M)$, is defined in terms of edge attributes and their semantics, and we also assume it to be normalized in $[0, 1]$. Both $c_N(a_D, a_M)$ and $c_E(e_D, e_M)$ are very particular to the problem and are defined after selecting the type of attributes to be used.

The next step is to define a fitness function to evaluate each of the solutions generated by the graph matching algorithm in order to find the best homomorphism h which satisfies the conditions $h : V_D \to V_M$ and $\forall a_M \in V_M \ \exists a_D \in V_D \ \mid \ h(a_D) = a_M$ as well as structural and similarity constraints.

3.4.2 f_1: only taking into account the matched vertices

Based on the definitions of the vertex similarity $c_N(a_D, h(a_D))$ and the edge similarity $c_E((a_D^i, a_D^j), (h(a_D^i), h(a_D^j)))$, the first proposal for a fitness function is to define the global similarity of a homomorphism h as:

$$f_1(h) = \frac{\alpha}{|V_D|} \sum_{a_D \in V_D} c_N(a_D, h(a_D)) +$$

$$\frac{1-\alpha}{|E_D|} \sum_{(a_D^i, a_D^j) \in E_D} c_E((a_D^i, a_D^j), (h(a_D^i), h(a_D^j))) \tag{3.1}$$

where $0 \leq \alpha \leq 1$ is a parameter used for tuning the relative importance of vertex and edge similarity.

3.4.3 f_2: taking into account all the vertices and similarities

The fitness function described in the previous section only considers similarity between vertices (or edges) that are mapped by the homomorphism h, but it does not take into account the fact that unmapped vertices could also have a high similarity which may be not desirable. Authors in [Perchant and Bloch, 1999] adopt this idea and propose a fitness function for the graph matching problem applied to medical images of the brain that has been used latter in [Perchant et al., 1999] and [Boeres et al., 1999]. This second proposal for a global similarity function accounting for such situations is presented in the following way:

$$f_2(h) = \frac{\alpha}{|V_D||V_M|} \sum_{(a_D,a_M)\in V_D\times V_M} [1 - |\theta_N(a_D,a_M) - c_N(a_D,a_M)|] +$$
$$\frac{1-\alpha}{|E_D||E_M|} \sum_{(a_D^i,a_D^j)\in E_D,(a_M^k,a_M^l)\in E_M} [1 - |\theta_E((a_D^i,a_D^j),(a_M^k,a_M^l)) - c_E((a_D^i,a_D^j),(a_M^k,a_M^l))|], \tag{3.2}$$

where

$$\theta_N(a_D,a_M) = \begin{cases} 1 & \text{if } a_M = h(a_D), \\ 0 & \text{otherwise,} \end{cases}$$

$$\theta_E((a_D^i,a_D^j),(a_M^k,a_M^l)) = \begin{cases} 1 & \text{if } a_M^i = h(a_D^k) \text{ and } a_M^j = h(a_D^l), \\ 0 & \text{otherwise,} \end{cases}$$

and $0 \leq \alpha \leq 1$ is a parameter used to adapt the weight of vertex and edge correspondences in $f_2(h)$. The fitness function can be easily understood by having a look to the two main terms: the first one measures the correspondence between vertices of the model and data graphs, and the second the correspondence between edges of both graphs. The term $\theta_N(a_D,a_M)$ is used to check whether two vertices are matched in the solution or not. If the two vertices are very similar ($c_N(a_D,a_M) \approx 1$) and the matching is not present in the solution that is being evaluated ($\theta_N(a_D,a_M) = 0$) then we will obtain for the matching a low value for this part of the represented solution ($[1 - |\theta_N(a_D,a_M) - c_N(a_D,a_M)|] \approx 0$). On the other hand, if this correspondence is present in the solution ($\theta_N(a_D,a_M) = 1$), we obtain a good value for this particular matching of the two vertices ($[1 - |\theta_N(a_D,a_M) - c_N(a_D,a_M)|] \approx 1$). The second term follows an analogous approach analyzing all the edges of both graphs.

3.4.4 f_3: not considering edges of vertices in G_D matched to the same vertex in G_M

Finally, a last improvement to the fitness function can be done by not taking into account edges of G_D between vertices that have been matched to the same vertex in G_M since the attributes of these edges are not meaningful and do not have to be compared to any edge of G_M. This idea to improve fitness functions is proposed and tested experimentally in [Boeres, 2002]. For that purpose, denote by $E_D^* = \{e_D = (a_D^i,a_D^j) \in E_D \mid h(a_D^i) \neq h(a_D^j)\} \subset E_D$ the set of edges that will only be considered for computing the global similarity. As a result, a third global similarity function is proposed:

$$f_3(h) = \frac{\alpha}{|V_D||V_M|} \sum_{(a_D,a_M)\in V_D\times V_M} [1 - |\theta_N(a_D,a_M) - c_N(a_D,a_M)|] +$$

$$\frac{1-\alpha}{|E_D^*||E_M|} \sum_{(a_D^i,a_D^j)\in E_D^*,(a_M^k,a_M^l)\in E_M} [1-|\theta_E((a_D^i,a_D^j),(a_M^k,a_M^l))-c_E((a_D^i,a_D^j),(a_M^k,a_M^l))|], \tag{3.3}$$

where α, $\theta_N(a_D,a_M)$ and $\theta_E((a_D^i,a_D^j),(a_M^k,a_M^l))$ are defined as in the previous section.

3.4.5 f_4: function based on the divergence between distributions

We propose the use of attributes based on probability theory, as these have not yet been applied to our graph matching problems. These require again to construct a completely different model and to represent it in a graph format using probabilistic parameters that will allow comparison of similarities with a data graph. Later on this chapter, two new types of fitness functions based on probability theory will be introduced.

The attributes that we will use can be classified in vertex attributes (unary attributes) and edge attributes (binary attributes) in the same way as described in Section 3.4.1.

The unary attribute that we consider in our examples for each of the vertices is the grey level distribution of the region represented by the vertex. Apart from this, there are also other three vertex attributes that will be used for other purposes but for measuring the similarity directly. These attributes are:

- size of the region (in pixels),
- coordinates x and y of the center of gravity of the region,
- super-region number in which the region is located, which is an attribute given by the tracking procedure used to find approximate landmarks of interesting features.

Analogously, the binary (edge) attributes that we will consider are:

- distance,
- relative position.

However, edge attributes will be represented in the form of vectors considering the center of gravity of the destination, exactly as done in [Cesar et al., 2002b]. Vectors will be computed from all the points of the origin to that destination-center of gravity, and we will calculate the mean and variance of the x and y components of all these vectors. Using this type of representation will implicitly record information of the distance and relative position.

Foundations of the new fitness functions based on probability theory

In order to consider attributes using probability, we will assume that each attribute of a region, either if it is a vertex or edge attribute, is modelled by a random variable that follows a normal distribution. For computing the similarity, we propose that both the model and data graphs are complete ones[3].

A vertex attribute such as the grey level can usually be fitted by a normal distribution (although this fact depends on the type of the input image). However, attributes such

[3]The generic definition of a complete graph is a graph $G=(V,E)$ such that $\forall a,a' \in V\ \exists e=(a,a')\in E$, and usually the condition $a \neq a'$ is also assumed. Therefore the edges from a vertex to itself are not considered. Here we assume that a complete graph does not contain such edges, with each vertex in a complete graph containing $|V|-1$ edges.

as distance, relative position, and in general, others more dependent on the shape of the region, could lead to problems when searching for a good solution. The modelization of these attribute values by normal distributions is more suitable when regions have a rounded shape.

However, when evaluating a solution, regions matched to a same model vertex can be considered as *fused* in a sense, and this fused region will often not have a rounded shape. As a result, and in order to allow a satisfactory approximation to normal distributions, it is very important to analyze carefully how to model edge attributes in both the model and data graphs.

The fact of using a representation based on vectorial components of the edge attributes is a solution to this problem. Both the distance and relative position attributes are implicit in a vectorial representation, as the distance corresponds to the modulus and the relative position to the angle of the vector.

In previous attributes based on fuzzy set theory that can be found in the literature such as [Perchant, 2000], these edge attributes are modelled using the necessity, and possibility (i.e. roughly equivalent to the minimum and the maximum). In our case, we will assume that attributes can be modelled by normal distributions, and therefore we will use the mean and variance.

The edges will be assumed to always finish in the center of gravity of the destination region. Therefore, the center of gravity of every region in both the model and data images will need to be computed. Each edge will be stored using their x and y components. As both the model and data graphs are complete ones, we will have $|V_i| - 1 \; i = M, D$ edges for each region respectively, where we will use two attributes for each.

As a result, for each vertex in both graphs we will consider the following attributes:

1. One unary attribute: the grey level distribution. As this attribute is assumed to follow a normal distribution, in a region a it will be represented as $\mathcal{N}_{a;g}(\mu_{a;g}, \sigma^2_{a;g})$, where μ is the mean and σ^2 is the variance.

2. Two edge attributes for each edge starting in the region a, and arriving at region k: $\mathcal{N}_{a,k;x}(\mu_{a,k;x}, \sigma^2_{a,k;x})$ and $\mathcal{N}_{a,k;y}(\mu_{a,k;y}, \sigma^2_{a,k;y})$. As in a complete graph with n vertices we have $n-1$ edges from each vertex, we have a total of $2n-2$ edge attributes for each vertex. Note that these definitions assume implicitly that the x and y components are independent, and therefore the variance-covariance matrix will be of the form $\begin{pmatrix} \sigma^2_{a,k;x} & 0 \\ 0 & \sigma^2_{a,k;y} \end{pmatrix}$. Another possibility is to consider a binary attribute by means of a 2-dimensional normal distribution, in which case the variance-covariance matrix would not contain zeroes.

Computing the attributes of each region

As attributes of the regions will be modelled by normal distributions, each attribute will be characterized by its mean and variance values. Therefore, in an image with n regions, a region a will be represented as an d-dimensional normal distribution, $\mathcal{N}_a(\boldsymbol{\mu}_a, \Sigma_a)$ (one vertex attribute and $2n-2$ edge ones, which makes $d = 2n-1$):

$$\begin{aligned}\mathcal{N}_a(\boldsymbol{\mu}_a, \Sigma_a) = \quad (\quad & \mathcal{N}_{a;g}(\mu_{a;g}, \sigma^2_{a;g}), \\ & \mathcal{N}_{a,1;x}(\mu_{a,1;x}, \sigma^2_{a,1;x}), \mathcal{N}_{a,1;y}(\mu_{a,1;y}, \sigma^2_{a,1;y}), \quad \ldots, \\ & \mathcal{N}_{a,a-1;x}(\mu_{a,a-1;x}, \sigma^2_{a,a-1;x}), \mathcal{N}_{a,a-1;y}(\mu_{a,a-1;y}, \sigma^2_{a,a-1;y}), \\ & \mathcal{N}_{a,a+1;x}(\mu_{a,a+1;x}, \sigma^2_{a,a+1;x}), \mathcal{N}_{a,a+1;y}(\mu_{a,a+1;y}, \sigma^2_{a,a+1;y}), \quad \ldots, \\ & \mathcal{N}_{a,n;x}(\mu_{a,n;x}, \sigma^2_{a,n;x}), \mathcal{N}_{a,n;y}(\mu_{a,n;y}, \sigma^2_{a,n;y}) \ \ ) \end{aligned} \tag{3.4}$$

This representation will be used for regions in the model and data graphs. It is important to take into account that the number of vertices in the model graph ($|V_M|$) is smaller than the number of vertices in the data graph ($|V_D|$). Each vertex in the model graph has $2|V_M|-1$ attributes and each vertex in the data graph has $2|V_D|-1$ attributes.

Cases in which mixtures of normal distributions are required

Mixtures of normal distributions are a way of representing a new distribution composed by a weighted sum of many normal distributions. Mixtures are used when the composition of two normal distributions cannot be approximated by a single new normal distribution. The later case occurs typically when the means of the different normal distributions are very far from each another, and therefore a weight is given to each of the distributions in order to express the contribution of each of them to the global combined distribution.

The weights of each original normal distribution are computed using algorithms such as the EM [Dempster et al., 1977]. This procedure is obviously very CPU expensive. In addition, comparing a mixture of normal distributions and a normal distribution (i.e. when comparing the solution proposed by an individual and a region in the model) is a lot more complicated than comparing two normal distributions in terms of number of operations required.

In our particular example, as proved on the following section, the fact of representing edge attributes as the two components (x and y) of a vector will allow us to approximate a combination of normal distributions to a new normal distribution, without having the need to use mixtures.

Attributes of the fused regions modelled by normal distributions

Let us consider as an example the case illustrated in Figure 3.4. This figure shows on the left two regions (A and B) of a model image, and on the right we have a typical result of an over-segmentation procedure, where the regions detected on a data image have lead to a data graph with more subregions for each model region. Therefore each correspondence of the regions in the model appears usually subdivided in the data image (the region A in the model has been divided in three subregions after the automatic over-segmentation procedure, regions 1, 2, and 3, and region B has been subdivided in two, regions 11 and 12).

In order to evaluate a solution, we would require to *fuse* regions 1, 2, and 3 in the data image, and then compare the similarity to the model region A. Similarly, data image regions 11 and 12 should be fused and afterwards compared to the model region B. The fusion of attributes of vertices in G_D such as the grey level (i.e. the one selected for our problems) represented as a normal distribution can be approximated to a new normal distribution for evident reasons, and it does not require further discussion.

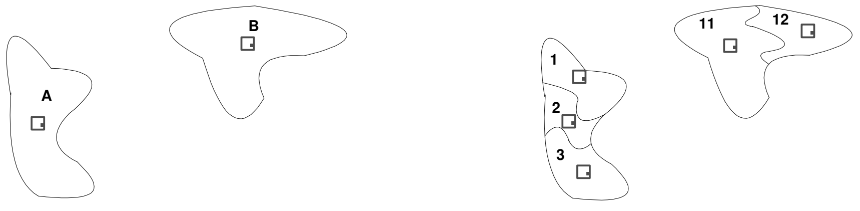

Figure 3.4: Illustration of two regions in the model graph (left), and the typical result after following an over-segmentation process on an image to be recognized (right). This figure also illustrates the centers of gravity of each of the regions. These will be used as a destination point representative of the whole vector from any point of the origin to the destination.

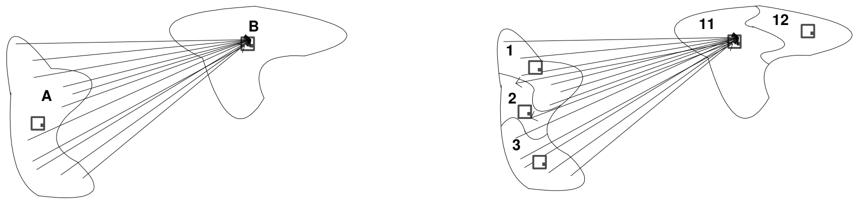

Figure 3.5: Example showing how the edge attributes are computed in both the model and data graphs.

On the other hand, as already explained, the edge attributes that are of our interest for comparing the similarity between regions are the distance and relative position. Given a homomorphism h, and in order to evaluate it, before starting with any comparison the regions in the data image that are matched to a same vertex in the model in h have to be fused, as they are supposed to be part of a same region. Figure 3.4 shows which centers of gravity that will be used for each region, and Figure 3.5 is an illustrative example on how this computation is done.

From the previous figures it is obvious that in the data graph we have edge attributes that are computed between subregions, but the equivalent regions in the model have a different center of gravity, and therefore the solution that can be obtained is also different. The proof that remains to be performed is illustrated in Figure 3.6, which essentially consists in showing that it is possible to compute the edge attributes between the model regions A and B starting from the edge attributes in the data graph of the data regions 1, 2, 3, 11 and 12. Next, we prove that this computation is possible and how it can be performed. In addition, we show that the *fusion* of edge attributes in the regions 1 2 and 3 into one, as well as such of the regions 11 and 12 into another, can effectively be modelled by a normal distribution using the vector components as representative data.

We will take as an example for this proof the regions and their centers of gravity of Figure 3.4. Let S_B be the size (in pixels) of region B in the model. Analogously, let S_{11} and S_{12} be the sizes of regions 11 and 12 respectively in the data graphs. As these two regions in the data graph are the equivalent as region B in the model graph, we assume that $S_B \equiv S_{11}+S_{12}$. Using geometrical properties it can easily be proved the x and y coordinates of the center of gravity of model region B $-B^B = \left(B_x^B, B_y^B\right)-$ can be computed from the

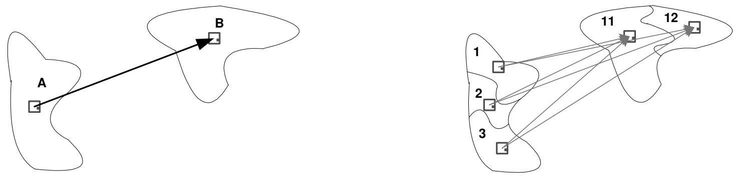

Figure 3.6: Summary of the problem of obtaining the edge attributes between regions A and B in the model graph, knowing the values of the edge attributes in the data graph.

centers of gravity of data image regions 11 and 12 –$B^{11} = \left(B_x^{11}, B_y^{11}\right)$ and $B^{12} = \left(B_x^{12}, B_y^{12}\right)$.

This result can be easily generalized for a model region R which is divided in m smaller subregions $r_1, r_2, \ldots, r_m$ in the data image with respective sizes in pixels $S_{r_1}, S_{r_2}, \ldots, S_{r_m}$. Then, it can be proved that the fused center of gravity of region R can be computed as

$$B^R = \left(B_x^R, B_y^R\right) \equiv \left(\frac{\sum_{i=1}^m B_x^{r_i} \cdot S_{r_i}}{\sum_{i=1}^m S_{r_i}}, \frac{\sum_{i=1}^m B_y^{r_i} \cdot S_{r_i}}{\sum_{i=1}^m S_{r_i}} \right) \tag{3.5}$$

Similarly, we can also prove that the edge attributes of any model region can also be computed starting from the edge attributes of the equivalent data image regions. Given a destination model region R divided in m smaller subregions $r_1, r_2, \ldots, r_m$ in the data graph with respective sizes $S_{r_1}, S_{r_2}, \ldots, S_{r_m}$, knowing that the fused center of gravity is the one described previously, and that $S_B \approx S^* = \sum_{i=1}^m S_{r_i}$, it can be easily shown that from any point $p_1 = (x_1, y_1)$ within the origin regions the following is satisfied:

$$\overrightarrow{v_{p_1 B^B}} \approx \sum_{i=1}^m \frac{S_{r_i}}{S^*} \cdot \overrightarrow{v_{p_1 B^{r_i}}} \tag{3.6}$$

This results is a linear combination of normal distributions. As such a combination of normal distributions is also a normal distribution, then we have proved that the when fusing the edge attributes of the $r_1, r_2, \ldots, r_m$ regions leads also to a region which edge attributes follow a normal distribution.

Evaluating a solution using the new approach

If individual regions of the data graph are to be compared to regions in the model graph, we could use one of these methods:

- The distance –or divergence– between both distributions could be used for comparison.
- Another possibility is to use the likelihood function to evaluate the similarity between regions.

We tested the use of both methods and compared them. Next, all the explanations and further formalization of these two methods are presented.

This fourth proposal of fitness function is based on the Kullback-Leibler divergence [Kullback and Leibler, 1951], which measures the difference between two different distributions. The Kullback-Leibler divergence is defined for any two types of distributions, although it can be written in a simple form if the two distributions to be compared are normal ones.

As explained in previous section, we will assume that all the attributes of all the regions in both the model and data graph follow normal distributions, and this assumption will help us to perform simpler operations to compute the divergence.

However, it is compulsory for this divergence that the dimensions of both normal distributions to be compared are the same. In fact, each of the regions of both the atlas and data graphs will be represented as a $(2n-1)$-dimensional normal distribution, where this dimension is $(2|V_M|-1)$ and $(2|V_D|-1)$ for each vertex of the model or data graphs respectively. Therefore, given a homomorphism, the question is how to find a way of comparing model and data vertices taking into account these restrictions for comparison. This comparison will be used as a similarity value.

In essence, when evaluating a solution given by a homomorphism h, each of the attributes of the regions in the data graph that are matched to the same vertex in the homomorphism (i.e. regions $a_D^i, a_D^j, \ldots, a_D^z \in V_D | h(a_D^i) = h(a_D^j) = \cdots = h(a_D^z) = a_M^r \in V_M$) have to be combined in a way that the combination of the normal distributions representing each attribute is approximated to a new normal distribution. This combination of regions will be done by *fusing* the regions which are labelled with the same model graph, in other words, the k^{th} fused region a_k^F is defined in the following way: $a_k^F = \{a_D^l \in V_D, k \in V_M | h(a_D^l) = a_M^k\}$.

It is important to note that after fusion of the data image regions and their attributes the result is a new region with $2|V_M|-1$ attributes.

After fusing all the data image regions to their corresponding *fused* region a_i^F $i = 1, \ldots, |V_M|$, their attributes are also combined, and this results in $2|V_M|-1$ attributes. Therefore, any fused region a^F can be represented as a new d-dimensional normal distribution $\mathcal{N}^F(\boldsymbol{\mu}^F, \Sigma^F)$ with $d = 2|V_M|-1$ (since there are 1 vertex attribute and $2|V_M|-2$ edge attributes) following the definition of $\mathcal{N}_a(\boldsymbol{\mu}_a, \Sigma_a)$ given in Equation 3.4 for any region a.

Afterwards, in order to compare how similar are the regions of the model and those of the data image we will only need to compare the d-dimensional normal distribution of the fused region a_k^F and the d-dimensional normal distribution of the model region a_M^k, with $d = (2|V_M|-1)$ in both graphs and $k = 1, \ldots, |V_M|$.

The main requirement of this method to be applied is to prove that the vertex and edge attributes of the fused regions can be satisfactorily approximated to normal distributions, knowing that the attributes of each of the vertices in V_D follow a normal distribution. This proof was shown in Section 3.4.5.

The Kullback-Leibler divergence. Kullback and Leibler introduced a divergence measure that is known as the Kullback-Leibler divergence [Kullback and Leibler, 1951]. The idea behind a divergence is to have a measure to quantify the information quantity given by data to discriminate between one or another probability distributions. In the discrete domain, having a finite set of values $S = \{a_1, a_2, \ldots, a_n\}$, and if the P and Q probability distributions are given by $P(a_i) = p_i$ and $Q(a_i) = q_i$ $i = 1, 2, \ldots, n$, then the Kullback-Leibler divergence is expressed by

$$D_{K-L}(P, Q) = \sum_{i=1}^{n} p_i \log \frac{p_i}{q_i} \tag{3.7}$$

Regarding the continuous case, given two density functions $f(x)$ and $g(x)$ for the space χ, we have that $\frac{dP}{dQ}(x) = \frac{f(x)}{g(x)}$, and therefore the Kullback-Leibler divergence is given by

$$D_{K-L}(P, Q) = \int_{\chi} f(x) \log \frac{f(x)}{g(x)} dx \tag{3.8}$$

In the case of having any two normal distributions of dimension d denoted by $\mathcal{N}(\boldsymbol{\mu}_1, \Sigma_1)$ and $\mathcal{N}(\boldsymbol{\mu}_2, \Sigma_2)$ respectively, the Kullback-Leibler divergence is computed in [Kullback and Leibler, 1951] as

$$\begin{aligned} D_{K-L}\left(\mathcal{N}(\boldsymbol{\mu}_1, \Sigma_1), \mathcal{N}(\boldsymbol{\mu}_2, \Sigma_2)\right) &= \frac{1}{2}\left[(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^t \Sigma_2^{-1} (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)\right] \\ &+ \frac{1}{2}\left(trace\left(\Sigma_2^{-1}\Sigma_1 - I\right) + \log\frac{|\Sigma_2|}{|\Sigma_1|}\right) \end{aligned} \quad (3.9)$$

where I is the unary matrix. This expression is further simplified when the elements of both d-dimensional normal distributions are independent. In the latter case, the matrices $\Sigma_j \;\; j = 1, 2$ of dimensions $d \times d$ would be of the form $\begin{pmatrix} \sigma_{j,1}^2 & 0 & 0 & \dots & 0 \\ 0 & \sigma_{j,2}^2 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & \sigma_{j,d}^2 \end{pmatrix}$. It can be proved that under these conditions the Kullback-Leibler distance is given by:

$$\begin{aligned} D_{K-L}(\mathcal{N}(\boldsymbol{\mu}_1, \Sigma_1), \mathcal{N}(\boldsymbol{\mu}_2, \Sigma_2)) &= \frac{1}{2}\left[\sum_{i=1}^{d} \frac{(\mu_1^i - \mu_2^i)^2}{\sigma_2^i}\right] \\ &+ \frac{1}{2}\left(\sum_{i=1}^{d}\left(\frac{\sigma_1^i}{\sigma_2^i} - 1\right) + \log\left(\prod_{i=1}^{d}\frac{\sigma_2^i}{\sigma_1^i}\right)\right) \end{aligned} \quad (3.10)$$

In our case, given a solution, for each of the model regions a_M^k two d-dimensional normal distributions with $d = 2|V_M| - 1$ will be compared: The normal distribution of the model vertex itself and the one obtained after fusing all the data regions associated to the k^{th} fused region a_k^F. Note also that the Kullback-Leibler divergence is not symmetrical, and that $D_{K-L}(P, Q) \neq D_{K-L}(Q, P)$. Therefore, as a reference has to be defined, we decided to choose as the reference the data graph vertex that is compared to each of the vertices of the model graph (note that the fitness function is also defined as $f_4(h) : V_D \to V_M$). Therefore, we consider the $D_{K-L}\left(\mathcal{N}\left(\boldsymbol{\mu}_{a_M^k}, \Sigma_{a_M^k}\right), \mathcal{N}\left(\boldsymbol{\mu}_{a_k^F}, \Sigma_{a_k^F}\right)\right)$ values for all the model regions.

Definition of the fitness function. Taking all the previous explanations into account, we propose a first fitness function definition based on the Kullback-Leibler distance. In this definition, denoted as f_4, when computing a solution proposed, we first *fuse* the regions in the data graph, and then we compare each of the vertex and edge attributes of the fused region to the vertex of the matched region. Taking the illustrative example of the previous section, the similarity between region A in the model and the combination of regions 1, 2, and 3 in the data graph will be computed using their respective d-dimensional normal distributions $\mathcal{N}(\boldsymbol{\mu}_A, \Sigma_A), \mathcal{N}(\boldsymbol{\mu}_1, \Sigma_1), \mathcal{N}(\boldsymbol{\mu}_2, \Sigma_2), \mathcal{N}(\boldsymbol{\mu}_3, \Sigma_3)$, with the sizes in pixels of each of the regions being S_A, S_1, S_2, and S_3 respectively. For this, the last three distributions have to be combined. Knowing that $\Sigma_j \;\; j = 1, 2, 3$ would be of the form $\begin{pmatrix} \sigma_{j,1}^2 & 0 & 0 & \dots & 0 \\ 0 & \sigma_{j,2}^2 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & \sigma_{j,d}^2 \end{pmatrix}$, we will denote the resultant distribution by $\mathcal{N}(\boldsymbol{\mu}_{123}, \Sigma_{123})$, having:

$$\boldsymbol{\mu}_{123} = \frac{S_1 \cdot \boldsymbol{\mu}_1 + S_2 \cdot \boldsymbol{\mu}_2 + S_3 \cdot \boldsymbol{\mu}_3}{S_1 + S_2 + S_3}$$

$$\Sigma_{123} = \begin{pmatrix} \tau^2_{123,1} & 0 & 0 & \dots & 0 \\ 0 & \tau^2_{123,2} & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & \tau^2_{123,d} \end{pmatrix} \tag{3.11}$$

where $\tau^2_{123,i} = \frac{S_1 \cdot \sigma^2_{1,i} + S_1 \cdot (\mu_{123,i} - \mu_{1,i})^2 + S_2 \cdot \sigma^2_{2,i} + S_2 \cdot (\mu_{123,i} - \mu_{2,i})^2 + S_3 \cdot \sigma^2_{3,i} + S_3 \cdot (\mu_{123,i} - \mu_{3,i})^2}{S_1 + S_2 + S_3}$ $i = 1, \dots, d$, and $\boldsymbol{\mu}_{123} = (\mu_{123,1}, \mu_{123,2}, \dots, \mu_{123,d})$. Note that the dimension of both d-dimensional normal distributions is $d = 2|V_M| - 1$.

The component of this region in the proposed fitness function will be defined as follows:

$$\begin{aligned} f_4(h)_{a_M} &= D_{K-L}(\mathcal{N}(\boldsymbol{\mu}_{a_M}, \Sigma_{a_M}), \mathcal{N}(\boldsymbol{\mu}_{123}, \Sigma_{123})) \\ &= \frac{1}{2}\left[\sum_{i=1}^{d} \frac{(\mu_{a_M,i} - \mu_{123,i})^2}{\sigma^2_{123,i}}\right] + \frac{1}{2}\left(\sum_{i=1}^{d}\left(\frac{\sigma^2_{a_M,i}}{\sigma^2_{123,i}} - 1\right) + \log\left(\prod_{i=1}^{d} \frac{\sigma^2_{123,i}}{\sigma^2_{a_M,i}}\right)\right) \end{aligned} \tag{3.12}$$

And finally, the global fitness function will be computed as

$$f_4(h) = \sum_{a_M \in V_M} f_4(h)_{a_M} \tag{3.13}$$

Generalizing this fitness function for a region a^k_M of the model that is matched to the equivalent m smaller subregions $r_1, r_2, \dots, r_m$ in the data graph (i.e. $h(r_1) = h(r_2) = \dots = h(r_n) = R$) with respectively sizes $S_{r_1}, S_{r_2}, \dots, S_{r_m}$, we have firstly that the combination of all the subregions in the data graph is given by the d-dimensional normal distribution $\mathcal{N}(\boldsymbol{\mu}_{fused_{a^k_M}}, \Sigma_{fused_{a^k_M}})$, knowing that

$$\begin{aligned} \boldsymbol{\mu}_{fused_{a^k_M}} &= \frac{\sum_{i=1}^{m} S_{r_i} \cdot \boldsymbol{\mu}_{r_i}}{\sum_{i=1}^{m} S_{r_i}} \\ \Sigma^2_{fused_{a^k_M}} &= \begin{pmatrix} \tau^2_{fused_{a^k_M},1} & 0 & 0 & \dots & 0 \\ 0 & \tau^2_{fused_{a^k_M},2} & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & \tau^2_{fused_{a^k_M},d} \end{pmatrix} \end{aligned} \tag{3.14}$$

where

$$\tau^2_{fused_{a^k_M},i} = \frac{\sum_{j=1}^{d}\left(S_{r_j} \cdot \sigma^2_{r_j,i} + S_{r_j} \cdot \left(\mu_{fused_{a^k_M},j} - \mu_{r_i,j}\right)^2\right)}{\sum_{j=1}^{d} S_{r_j}}$$

for $i = 1, \dots, d$, and $\boldsymbol{\mu}_{fused_{a^k_M}} = \left(\mu_{fused_{a^k_M},1}, \mu_{fused_{a^k_M},2}, \dots, \mu_{fused_{a^k_M},d}\right)$. Finally, the fitness function is defined as

$$f_4(h) = \sum_{a^k_M \in V_M} f_4(h)_{a^k_M} = \sum_{a^k_M \in V_M} D_{K-L}\left(\mathcal{N}\left(\boldsymbol{\mu}_{a^k_M}, \Sigma_{a^k_M}\right), \mathcal{N}\left(\boldsymbol{\mu}_{fused_{a^k_M}}, \Sigma_{fused_{a^k_M}}\right)\right) \tag{3.15}$$

As this fitness function expresses the divergence of probability distributions, in this case the graph matching algorithm is committed to return the individual that minimizes this expression, and not to maximize as in the previous three fitness functions.

3.4.6 f_5: function based on likelihood

This fifth proposal of fitness function is based on the likelihood of the data given the model. In this case, the regions in the model are also represented as a d-dimensional normal distribution, for a dimension for each of the vertex and edge attributes, as well as for the previous fitness function $f_4(h)$: the mean and variance of each of the attributes for all the regions of the model are computed as a previous step, and these are used for computing the likelihood. Regarding the data graph, a similar representation is assumed, and also the data information required to compute the likelihood is stored. This information is later used to compute each of the homomorphisms.

For this approach, we select at random a set of N pixels from each of the regions in the data graph. We decided to choose this number N as a constant for all type of regions independently of their size, although when fusing the regions in the data image only a number of pixels proportional to the size of the fused region are taken later into account. The number N of pixels selected has to be big enough to ensure representation of each of the regions in the data image. We denote the representation of pixels for the k^{th} region of the data image by x_1^k, x_2^k, ..., x_N^k , with $k = 1, \ldots, |V_D|$. The total size of this sample is $M = N|V_D|$.

The next step is to compute the attribute values of the selected pixels in each region. In the case of the vertex attributes, we consider again the grey level and the texture value. For the edge attributes, we compute again the binary attributes using as a reference the pointed vertex the center of gravity of the regions that have to be fused in the homomorphism as explained in the previous sections. As a result, the number of attributes to be recorded by each sample pixel is $2|V_D| - 1$. The attributes for the i^{th} pixel of the k^{th} region is denoted by $\boldsymbol{y}_i^k = \left(y_{i,1}^k, y_{i,2}^k, \ldots y_{i,2|V_D|-1}^k\right)$.

Computing the likelihood of the data regarding each attribute. Given the probability distribution of a single attribute j of a particular region k of the model, $\mathcal{N}(\mu_{k,j}, \sigma_{k,j}^2)$, the likelihood is a measurement with N possible values $y_{1,j}^k, y_{2,j}^k, \ldots, y_{N,j}^k$ defined as:

$$\begin{aligned} L_j(y_{1,j}^k, y_{2,j}^k, \ldots, y_{N,j}^k; \mathcal{N}(\mu_{k,j}, \sigma_{k,j}^2)) &= \prod_{i=1}^{N} \left[\frac{1}{\sqrt{2\pi}\ \sigma_{k,j}} \exp\left(-\frac{1}{2\sigma_{k,j}^2} (y_{i,j}^k - \mu_{k,j})^2 \right) \right] \\ &= \left(\frac{1}{\sqrt{2\pi}\ \sigma_{k,j}} \right)^N \exp\left(-\frac{1}{2\sigma_{k,j}^2} \sum_{i=1}^{N} (y_{i,j}^k - \mu_{k,j})^2 \right) \end{aligned} \tag{3.16}$$

In order to evaluate a given homomorphism h, we proceed in a similar way as in the previous sections: firstly, the regions in the data image following the labels given by the homomorphism h are identified, and $|V_M|$ fused regions are obtained. Secondly, the centers of gravity of these fused regions are computed, and the vertex and edge attributes are recomputed. As a result, each of the fused regions contains 1 vertex attribute and $2|V_M| - 2$ edge attributes. Thirdly, a set of N pixels are selected from each of the fused regions. Fourthly, the likelihood of the d-dimensional probability distribution representing a model region $a_M^k \in V_M$, with $d = 2|V_M| - 1$, is computed as:

$$L\left(\boldsymbol{y}_1^k, \boldsymbol{y}_2^k, \ldots, \boldsymbol{y}_N^k; \mathcal{N}\left(\boldsymbol{\mu}_{a_M^k}, \Sigma_{a_M^k}\right)\right) = \prod_{j=1}^{2|V_M|-1} L_j(y_{1,j}^k, y_{2,j}^k, \ldots, y_{N,j}^k; \mathcal{N}(\mu_{k,j}, \sigma_{k,j})) \tag{3.17}$$

where L_j is defined in Equation 3.16. The result obtained in Equation 3.17 is used in $f_5(h)$ to compute the similarity between region k of the model and the k^{th} fused region in the data image.

Evaluation of a homomorphism: definition of the fitness function. Following the definitions of the previous section, a homomorphism h can be evaluated by means of the likelihood of the models for every vertex in the model graph regarding the pixels sampled in the data image. For that, the similarities of all the vertices of the model graph with such of the corresponding fused region in the data image are considered to define the global similarity –GS– of a given homomorphism h as follows:

$$\begin{aligned} GS &= \prod_{k=1}^{|V_M|} L\left(\boldsymbol{y}_1^k, \boldsymbol{y}_2^k, \ldots, \boldsymbol{y}_N^k; \mathcal{N}\left(\boldsymbol{\mu}_{a_M^k}, \Sigma_{a_M^k}\right)\right) \\ &= \prod_{k=1}^{|V_M|} \prod_{j=1}^{2|V_M|-1} L_j(y_{1,j}^k, y_{2,j}^k, \ldots, y_{N,j}^k; \mathcal{N}(\mu_{a_M^k,j}, \sigma_{a_M^k,j})) \\ &= \prod_{k=1}^{|V_M|} \prod_{j=1}^{2|V_M|-1} \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi}\ \sigma_{a_M^k,j}} \exp\left(-\frac{1}{2\sigma^2_{a_M^k,j}}(y_{i,j}^k - \mu_{a_M^k,j})^2\right) \\ &= \left[\prod_{k=1}^{|V_M|} \prod_{j=1}^{2|V_M|-1} \left(\frac{1}{\sqrt{2\pi}\ \sigma_{a_M^k,j}}\right)^N\right] \exp\left(\sum_{k=1}^{|V_M|} \sum_{j=1}^{2|V_M|-1} \sum_{i=1}^{N} \left(-\frac{(y_{i,j}^k - \mu_{a_M^k,j})^2}{2\sigma^2_{a_M^k,j}}\right)\right) \end{aligned} \tag{3.18}$$

where $\mathcal{N}\left(\boldsymbol{\mu}_{a_M^k}, \Sigma_{a_M^k}\right)$ $k = 1, \ldots, |V_M|$ are the ones defined in the previous sections, and all the $\boldsymbol{y}_1^k, \boldsymbol{y}_2^k, \ldots, \boldsymbol{y}_N^k$ are the attribute values of the N pixels chosen at random from the fused region k, which is formed by the $a_D \in V_D$ such that $h(a_D) = k$.

However, in order to define the fitness function $f_5(h)$ using this similarity definition, the expression in Equation 3.18 can be simplified, since a proportional fitness function is valid for our purpose. A simpler yet proportional definition of this equation is described below:

$$GS = C\ \exp\left(\sum_{k=1}^{|V_M|} \sum_{j=1}^{2|V_M|-1} \sum_{i=1}^{N} \left(-\frac{(y_{i,j}^k - \mu_{a_M^k,j})^2}{2\sigma^2_{a_M^k,j}}\right)\right) \propto \sum_{k=1}^{|V_M|} \sum_{j=1}^{2|V_M|-1} \sum_{i=1}^{N} \frac{(y_{i,j}^k - \mu_{a_M^k,j})^2}{-2\sigma^2_{a_M^k,j}} \tag{3.19}$$

where $C = \left[\prod_{k=1}^{|V_M|} \prod_{j=1}^{2|V_M|-1} \left(\frac{1}{\sqrt{2\pi}\ \sigma_{a_M^k,j}}\right)^N\right]$. As a result, the fitness function $f_5(h)$ can be defined as the simple expression

$$f_5(h) = \sum_{k=1}^{|V_M|} \sum_{j=1}^{2|V_M|-1} \sum_{i=1}^{N} \frac{(y_{i,j}^k - \mu_{a_M^k,j})^2}{-2\sigma^2_{a_M^k,j}} \tag{3.20}$$

In this case, as well as with $f_1(h)$, $f_2(h)$, and $f_3(h)$, the graph matching algorithm is committed to return the individual that maximizes this expression.

3.5 Conclusion

This chapter presents the graph matching problem as a combinatorial optimization one, analyzing the main aspects that have to be taken into account for that: the definition of an individual representation and its associated fitness function. Three different individual representations have been presented in the discrete domain, as well as five different fitness functions.

For the continuous domain, an individual representation based on permutations has been introduced. This representation allows to apply the fitness functions defined for the discrete domain for specific graph matching algorithms in the continuous domain.

It is important to note that all the definitions given in this chapter can be applied by any combinatorial optimization algorithm. In the next chapters algorithms such as estimation of distribution algorithms, genetic algorithms, and evolutionary strategies apply these fitness functions to search for the best homomorphism.