Graph Classification Using Genetic Algorithm and Graph Probing Application to Symbol Recognition

Eugen Barbu, Romain Raveaux, Hervé Locteau, Sébastien Adam, Pierre Héroux, Éric Trupin

Graph Classification Using Genetic Algorithm and Graph Probing Application to Symbol Recognition

Eugen Barbu, Romain Raveaux, Hervé Locteau, Sébastien Adam, Pierre Héroux, Eric Trupin
LITIS Labs – University of Rouen, FRANCE
Eugen.Barbu@univ-rouen.fr

Abstract

We present in this paper a graph classification approach using genetic algorithm and a fast dissimilarity measure between graphs called graph probing. The approach consists in the learning of a set of synthetic graph prototypes which are used for a 1NN classification step. Some experiments are performed on real data sets, representing 10 symbols. These tests demonstrate the interest to produce prototypes instead of finding representatives which simply belong to the data set.

1. Introduction

Graphs are frequently used in various fields of computer sciences since they constitute a universal modelling tool which allows to describe structured data. The involved objects and their relations are described in a unique formalism. It is particularly the case in pattern recognition, and moreover in symbol recognition [1] since symbols can be naturally described using primitives (vectors, arcs, connected components, loops…) and geometric relations between these primitives (neighbourhood, connection, parallelism…). In such a case, the pre-segmented symbol recognition problem turns into a graph classification problem [2][3]. Its objective is to assign a graph describing an unknown symbol to its class using a learning database. In the context of symbol recognition, this database is generally reduced to one model per class (the ideal model) since models are generally issued from CAD systems.

In this paper, we describe a system able to classify graphs representing symbols using a learning database initially composed of one sample per class. Despite this lack of examples, our aim is to take into account the variability which can occur in symbol image representation. Hence, from a set of N ideal graphs describing the N symbol classes, our algorithm aims at learning a set of graph prototypes taking into account these possible distortions. Then, these prototypes are used in a classification step in order to determine the class of an unknown and noisy symbol in a recognition system. Our approach can be decomposed into 3 steps. First, a synthetic noise generation algorithm is applied on the ideal symbol model of each class in order to obtain a set of M graphs per class from the set of degraded symbols. This noise [4] has been proved to be representative of the variability which can occur on document images. Then, from this learning set, a graph based **G**enetic **A**lgorithm (GA) is applied. Its aim is to generate a set of K graph prototypes for each class. The value to be optimized by the GA is the recognition rate which is obtained in the simulation of a 1-NN classification algorithm using the selected prototypes as learning samples and a test database. Both steps (prototypes learning and classification) use a measure called *graph probing* in order to evaluate the similarity between graphs. This measure has been chosen after a comparative study between different approaches. Finally, in a validation step, a 1-NN classification algorithm is applied using the selected prototype set as learning elements, a validation database and the same dissimilarity measure.

The paper is organised as follows: In the second section, the graph probing concept is introduced. Then, the third section presents the genetic algorithm in use, and particularly the specific genetic operators involved. The fourth section presents the application to the symbol recognition problem, the comparative study between the tested dissimilarity measures and the obtained classification results. Finally, a conclusion is given and future works are brought in section 5.

2. Dissimilarity measures

Measures of dissimilarity between complex objects which have a structure (sets, lists, strings, …) are based on the quantity of shared terms.

The simplest similarity measure between two objects is the matching coefficient, which is based on the number of common terms. Using this idea as a starting point, dissimilarity measures which take into account the maximal common subgraph (MCS) of two graphs were proposed in [5].

Another method which proposes a metric distance in the universal set of graphs is the edit distance. It represents the minimum-cost sequence of basic editing operations (e.g. insertion or deletion of vertices and edges with associated costs). The graph edit distance and MCS computation are equivalent to each other under a certain cost function associated to edit operations [6]. These distances between graphs have worst case exponential running times.

In our application, we use a genetic algorithm which employs intensively computations of dissimilarities between graphs. Hence, we have to find faster algorithms which compute dissimilarities, eventually approximations. In such a context, the graph topology can be partially ignored by considering an approximation which uses independently the set of vertices and arcs, for instance, edge matching distance or vertex matching distance [7]. Edge matching distance proposes a cost function for the matching of edges and then derives a minimal weight maximal matching between the edge sets of two graphs. This matching has a worst case complexity of $O(n^3)$, where n is the number of edges of the largest graph.

Another possibility to define a similarity measure is to count the number of occurrences of a set of sub graphs (named fingerprints or probes in different contexts) from each graph and to describe the objects to be compared as vectors [8]. In this setting (named graph probing), the similarity between graphs is the similarity between the two associated vectors. These methods are fast since they can be run in linear time, but they do not imply that if the distance between two graphs is 0 the graphs are isomorphic. However, a lower bound relation within a factor of four exists between the graph probing and the edit distance [8]. An experimental comparison between graph probing and other approaches is presented in section 4.

3. The genetic algorithm in use

Genetic **A**lgorithms (GAs) are adaptive heuristic optimisation algorithms based on the evolutionary ideas of natural selection and genetics. The basic concept of GAs is designed to simulate natural processes, necessary for evolution of artificial systems. They represent an intelligent exploitation of a random search within a defined search space to solve a problem. As can be seen on fig 1, after a random initialization of a population of possible solutions, GA's are based on a sequential ordering of four main operators: selection, replication, crossover and mutation. In order to apply genetic algorithms to a given problem, three main stages are necessary: the coding of the problem solutions, the definition of the objective function which attributes a fitness to each individual, and the definition of the genetic operators which promote the exchange of genetic material between individuals. In most existing GA applications, a linear representation of individuals is used. Problem parameters are encoded through a binary or a real string. Crossover is then applied through a single-point or two-point based exchange of genes. Regarding mutation, it is applied through a random modification of a small number of genes chosen randomly.

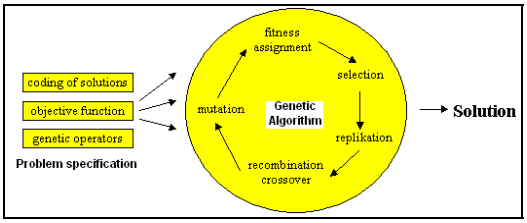

Fig 1: Overview of a genetic algorithm

In our context of pattern recognition using graph, each individual has to encode a set of graphs (the KxN prototypes). Consequently, the evolution of the individuals through GA implies to revisit classical operators since they have to modify graphs.

Concerning mutation, our operator is based on the six unary edit operations which can be applied to a graph: add or remove a node or an edge and modify a node or an edge label. For each mutation operation, we first decide to apply or not the operator according to a pre-defined rate. If mutation has to be applied, one of the six possibilities is chosen randomly, as well as the new label if the operation is a label modification.

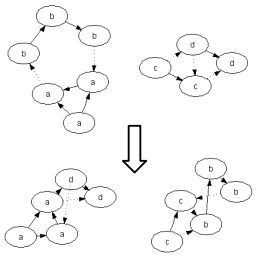

Fig 2: The crossover operator

To perform crossover between individuals (see fig. 2), we first randomly partition the set of nodes of each graph in two subsets (see the label of nodes on figure

2). We call *internal edges*, the edges of the initial graph the nodes of which are in the same subset (continuous lines). At the opposite, edges the nodes of which are in different subsets are called external edges (dotted lines). Then, a node is said to be an output node if it is a source of external edge, and an input node if it is the destination of an external edge. Finally, according to the nature of nodes and edges, fragments are swapped and edges are recombined so that all external edges now point to randomly selected input nodes.
Crossover and mutation are combined sequentially as shown in figure 1, after a classical selection process using a fitness based roulette wheel approach.

4. Application

In this section, the graph construction step is explained. Then, a comparative study concerning dissimilarity measure is described. It justifies our decision to use graph probing in our context. Finally, the symbol recognition application is presented, results are proposed and measured up to another approach.

4.1. Graph data set construction

Our data are made of graphs corresponding to a corpus of 180 symbol images, generated from 10 ideal models proposed in a symbol recognition contest (GREC workshop). In a first step, considering the symbol binary image, we extract both black and white connected components. These connected components are automatically labeled with a partitional clustering algorithm [10] applied on a set of features called Zernike moments [11]. Using these labeled items, a graph is built. Each connected component represents a attributed vertex in this graph. Then, edges are built using the following rule: two vertices are linked with an undirected and unlabeled edge if one of the node is one of the *h* nearest neighbours of the other node in the corresponding image. The values of the number of clusters *c* found by the clustering algorithm, and *h* the number of significant neighbours has been done after a comparative study. An example of the association between two symbol images and the corresponding graphs is illustrated in fig 3.

4.2. Test on dissimilarity distances

In order to choose the best dissimilarity measure in the context of our application, a study has been led concerning the correlation values between the dissimilarity measures proposed in section 2. Two experiences compose this study. First, we have computed Pearson correlation coefficients (cor) between the different dissimilarity measures. Results are presented on the first line of table 1. The second experience concerns the correlation between a user-defined ground truth order (or partial order) and the order calculated using the distance between representations. Such a correlation has to be as high as possible since our objective is the classification of graphs. This correlation can be measured using the Kendall rank correlation coefficient (tau) [9]. Using these values, we can select a graph representation and a dissimilarity measure which satisfies both running time constraints and high correlation with the ground-truth of our application. The obtained values, associated with the corresponding run time complexity, point out the trade-off to be made between the quality (agreement with the ground truth) of a similarity measure and its run time complexity. Since our application is quite demanding of dissimilarity measures, graph probing seems more suitable, showing a better trade-off: meaningful and operating in linear time.

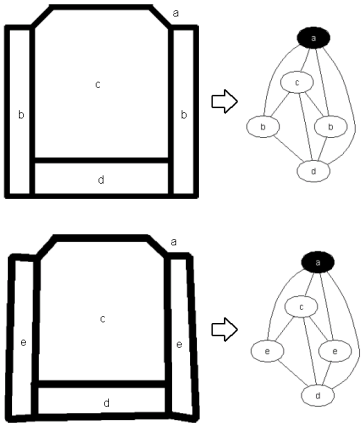

Fig 3: From symbols to Graphs

Table 1. Correlation between the edit distance (ED), the edge matching distance(EMD), graph probing(GP) and Ground Truth (GT).

	ED	GP	EMD
ED using cor	1	0.58	0.63
ED using tau	1	0.53	0.63
GT using tau	0.699	0,622	0,657

4.3. Classification experiments

As said before, the learning algorithm consists in the generation of K graph prototypes per symbol class

for a group of N classes. These prototypes are produced by a graph based GA, the aim of which is to find the near optimal solution of the recognition problem using the selected prototypes.

In such a context, each individual in our GA is a vector containing K graphs per class, that is to say K feasible solutions (prototypes) for a given class. Hence, an individual is composed of KxN graphs.

For the initialisation of the population, each graph of each individual is selected randomly from the initial graph corpus. The fitness (the suitability) of each individual is quantified thanks to the classification rate obtained using the corresponding prototypes and a test database. The classification is processed by a 1-NN classifier using the graph probing distance. Then, using the operators described in section 3, the GA iterates, in order to optimise the classification rate. The stopping criterion is the generation number.

At the end of the GA, a classification step is applied on a validation database in order to evaluate the quality of the selected prototypes. The obtained results are compared with an approach which also finds K representatives in a set of objects, described only by their reciprocal matrix distance. This approach which minimises (unsquared) distances from objects to representatives, is called Partition Around Medoids (PAM) [10]. PAM gives us its K best prototypes for a given class. Using these prototypes and the graph probing distance, we can compute the recognition rate. Table 2 gives the comparative results for K=1,2,3 prototypes per class, and a group of 10 classes. One can also note that using only the ideal models as learning set, a 1NN classification using graph probing provides a 88% recognition rate.

All these results show the interests of the prototype selection using GA combined with the graph probing approach. According to us, the main reason is that the learning application creates NxK synthetic elements thanks to the genetic operators in order to obtain the best representation of a particular class. Hence, our range of possibilities is not limited to the graphs constituting the class.

Table 2: Global classification rate

K	1	2	3
PAM	81,14%	95,22%	96,66%
GA	98,92%	99,17%	99,44%

5. Conclusion

In this paper, a graph classification algorithm has been proposed with an application to symbol recognition. The approach is based on the learning of graph prototypes using genetic algorithm and a fast dissimilarity measure called graph probing. This measure has been judged more efficient from the computation speed point of view. The obtained results, compared with a classification using PAM to select prototype, have shown the interest to generate synthetic prototypes through the use of genetic operators rather than finding them among the elements defining the classes. Our further works concern different points. The first of them consists in enriching the symbol description as graph, for example through the use of contour vectorisation results. A second one consists in testing the approach on a more important database. Another one consist in comparing the approach with the use of graph kernel SVM. Finally, a promising further work concerns the integration of reject in the system. In this way of thinking, a multi-objective approach is necessary. Its aim is to provides a wide range of solutions, structured as couples: (confusion rate, classification rate). Such solutions will guaranty the perfect fit for the system needs.

6. References

[1] J. Llados, E. Valveny, G. Sanchez and E. Marti, “Symbol recognition: Current advances and perspectives”, Lecture Notes in Computer Science, N° 2390, 2001, pp 104-127.

[2] A. Schenker, M. Last, H. Bunke and A. Kandel. "Classification of web documents using a graph model", In Proceedings of the 7th International Conference on Document Analysis and Recognition (ICDAR), 2003, pp 240-244.

[3] A. Serrau, G.L. Marcialis, H. Bunke and F. Roli, “An experimental comparison of fingerprint classification methods using graphs”, Lecture Notes in Computer Science, N° 3434, 2005, pp 281-290

[4] E. Valveny and Ph. Dosch. “Symbol Recognition Contest: A Synthesis”. Lecture Notes in Computer Science, N° 3088, 2004, pp. 368-385.

[5] H. Bunke and K. Shearer, “A graph distance metric based on the maximal common subgraph”. Pattern Recogn. Lett., 19, 1998, pp 255-259.

[6] H. Bunke, "On a relation between graph edit distance and maximum common subgraph", Pattern Recogn. Lett., 18, 1997, pp 689-694.

[7] H.P. Kriegel and S. Schönauer, “Similarity Search in Structured Data”, Lecture Notes in Computer Science, N° 2737, 2003, pp. 309-319.

[8] D. P. Lopresti and G.T. . Wilfong, “A fast technique for comparing graph representations with applications to performance evaluation”, International Journal on Document Analysis and Recognition, 6, 2003, pp 219-229.

[9] M. G. Kendall, “Rank Correlation Methods”, Hafner Publishing Co., New York, 1955.

[10] L. Kaufman and P.J. Rousseeuw, “Finding groups in data”, John Wiley & Sons, Inc., New York, 1990

[11] A. Khotazad and Y.H. Hong, “Invariant image recognition by Zernike Moments”, PAMI, Vol 12, No 5, May 1990, pp 489-497.