Automatic Classification of Graphs by Symbolic Histograms

Guido Del Vescovo and Antonello Rizzi, Member, IEEE
INFOCOM Department, University of Rome "La Sapienza"
Via Eudossiana, 18 - 00184, Rome, Italy
{rizzi, delvescovo}@infocom.uniroma1.it

Abstract

An automatic classification system coping with graph patterns with node and edge labels belonging to continuous vector spaces is proposed. An algorithm based on inexact matching techniques is used to discover recurrent subgraphs in the original patterns, the synthesized prototypes of which are called symbols. Each original graph is then represented by a vector signature describing it in terms of the presence of symbol instances found in it. This signature is called symbolic histogram. A genetic algorithm is employed for the automatic selection of the relevant symbols, while a K-nn classifier is used as the core inductive inference engine. Performance tests have been carried out using algorithmically generated synthetic data sets.

1. Introduction

A labeled graph is a very useful data structure when it comes to represent patterns which originate as sets of objects with some relationships between them. Each node label holds the descriptor of an object, whereas each edge label holds the descriptor of the relationship between two objects. Applications employing labeled graphs to represent patterns include scene analysis, classification of images containing discrete objects [1], chemical compounds [2], etc.

A graph classification problem consists in synthesizing a model of an unknown classification process on the basis of a training set S_{tr} containing graph patterns used as examples, such that a given performance measure is maximized over a test set S_{ts}. In the following we will call *instance* of a graph classification problem a given pair $< S_{tr}$, $S_{ts} >$.

An effective way to solve a graph classification problem consists in performing data mining tasks in order to detect frequent subgraphs in the entire training set. Once the frequent subgraphs are found and stored as prototypes in an alphabet, each pattern can be represented by a feature vector (*symbolic histogram*) describing it in terms of the number of instances of the alphabet's elements recognized in it. In [2] such a technique is employed to classify chemical compounds. The system proposed in the present paper adopts a similar technique to classify graphs with node and edge labels belonging to continuous vector spaces.

For that reason, and due to the presence of noise and/or distortions, exact matching techniques cannot be adopted to find frequent subgraphs and to recognize their instances in the test data [3]. Thus, a clustering technique is employed to detect compact and highly populated subgraph clusters which represent significant substructures. A hierarchical clustering algorithm is run with different metric parameter settings, in order to obtain substructure clusters characterized by individual metric weights. The maximum number of nodes of the substructures is kept low in order to avoid the abrupt increase in computational cost, which mostly depends on the search and matching procedures of subgraphs with a high number of nodes. As shown in the following, limiting the size of the substructures doesn't prevent the system to successfully solve classification problems defined on the basis of larger significant subgraphs, although the extraction of semantic information could be incomplete in such cases.

The symbolic histogram can be considered as a pattern representation by which graphs are mapped into R^n vectors. As a consequence, it is possible to employ well known inference engines for the actual classification. The proposed system adopts a *K–nn* classification technique. Since, usually, the frequent substructure search algorithm leads to the synthesis of a large number of symbols which are not related to the classification problem at hand, a genetic algorithm is in charge to select the relevant features of the symbolic histogram. We have specifically generated a set of synthetic classification problems with the aim to

0-7695-3032-X/07 $25.00 © 2007 IEEE
DOI 10.1109/GrC.2007.140

evaluate the automation degree of the system and its robustness to noise and distortions.

2. Definitions and notation

A *vector labeled graph* $g = (V, E)$ is constituted by a set of nodes V and by a set of edges E; each node holds a label belonging to R^h and each edge holds a label belonging to R^k. The features of the node and edge labels are logically grouped into *sections* [1], in order to easily take into account the different nature or meaning of each section in each processing step. The size of a graph g is its number of nodes $|V|$. The operator $|\ldots|$, here and in the following, denotes the cardinality of a discrete set or of a list of elements. A subgraph $g' = (V', E')$ of g is a graph such that $V' \subseteq V$ and E' contains all the edges in E connecting two nodes in V'.

An ECGM (see also [4]) $f : g^{(1)} \to g^{(2)}$ between two graphs $g^{(1)} = (V^{(1)}, E^{(1)})$ and $g^{(2)} = (V^{(2)}, E^{(2)})$ is a set of two bijective relations $f^V : V^{(1)} \to V^{(2)}$ and $f^E : E^{(1)} \to E^{(2)}$, such that:

$$\left.\begin{array}{l} f^V(v_i^{(1)}) = v_h^{(2)} \\ f^V(v_j^{(1)}) = v_k^{(2)} \end{array}\right\} \Rightarrow f^E(e_{ij}^{(1)}) = e_{hk}^{(2)}, \tag{1}$$

where $e_{ij}^{(1)}$ is the edge in $E^{(1)}$ connecting the nodes $v_i^{(1)}$ and $v_j^{(1)}$ in $V^{(1)}$. The dissimilarity measure between two graphs is defined as follows:

$$diss_\mu\left(g^{(1)}, g^{(2)}\right) = \min_{f \in \Psi}\left\{cost_\mu(f)\right\}, \tag{2}$$

where Ψ is the set of all the possible ECGM between $g^{(1)}$ and $g^{(2)}$. The cost of an ECGM is defined as in [1]. The ECGM which minimizes the cost is called the optimum ECGM. The same definition can be extended to the case of two subgraphs present on the same pattern or on two different patterns. As in [1], the dissimilarity measure is parametric with respect to the weights of the various sections (where each weight can be 0 or 1). The sub-index μ appearing in (2) expresses the dependency on the metric parameter set, exactly. In contrast to the system described in [1], the system presented here performs an exhaustive search to determine the optimum ECGM instead of a heuristic one, thanks to the limited size of the considered subgraphs in the symbol discovery step.

If $G = \{g_0, g_1, \ldots, g_{n-1}\}$ is a set of graphs (or subgraphs) all of the same size, the *graph centroid* $\tilde{g} = (\tilde{V}, \tilde{E})$ of the set can be defined and computed by the following algorithm:

```
$\tilde{g} = g_0$
for $i = 1$ to $n-1$
    1. compute the optimum ECGM $f_{opt} : \tilde{g} \to g_i$
    2. $\tilde{v} = \frac{i \cdot [\tilde{v} + f_{opt}^V(\tilde{v})]}{i+1}, \forall \tilde{v} \in \tilde{V}$
    3. $\tilde{e} = \frac{i \cdot [\tilde{e} + f_{opt}^E(\tilde{e})]}{i+1}, \forall \tilde{e} \in \tilde{E}$
```

A *symbol* is a prototype derived form a cluster C of subgraphs of equal size, collected over the entire data set. A symbol is constituted by the following data fields:

- The size d of the subgraphs contained in C.
- The centroid $\tilde{c}$ of C.
- The metric parameter set μ used to determine C.
- A threshold τ_R representing the maximum distance between $\tilde{c}$ and any graph or subgraph g of size d in order to recognize g as matching with the symbol.

3. Symbolic graph preprocessing

The symbolic preprocessing procedure consists in the two following steps:

1. The symbolic analysis is performed on the training set S_{tr} and an alphabet of symbols is generated.
2. The patterns contained in S_{tr} are represented by feature vectors describing them in terms of the presence of instances of the various symbols (symbolic histograms), yielding the preprocessed training set $\hat{S}_{tr}$.

The preprocessed training set $\hat{S}_{tr}$ is then used to train a classification model. In the test set classification stage, the test set S_{ts} is preprocessed with the same procedure used in the step 2 for S_{tr} on the basis of the same alphabet generated during the symbolic analysis of S_{tr}. The preprocessed test set $\hat{S}_{ts}$ is submitted to the classification model in order to test the generalization capability of the system. In the following subsections, each of the 2 steps described above will be illustrated in detail.

3.1. Symbol discovery (step 1)

In order to find the frequent subgraphs, lists of candidates of growing size are generated [3], and a clustering algorithm is run. Each cluster is a candidate symbol and is evaluated by a cost function. The algorithm is constituted by an initialization step and by a main loop.

Initialization

The current subgraph set Π, which is the bunch we search for clusters of similar elements, is filled with each subgraph of order 1 (i.e. single nodes) from the entire training set. A repository *M* is filled with a predetermined list of metric parameter settings. The content of *M* can be established on the basis of the set of problems to face. In the current implementation all the sets containing one or two non-zero weights are considered, thus $|M|$ depends on the number of sections in which node and edge feature vectors are subdivided. A temporary cluster list, *temp_list* and the symbol list *alphabet*, which is the output of the algorithm, are initialized as empty.

Main loop

```
while $size(\Pi) \leq d_{\max}$
  for $m = 0$ to $|M|-1$
    $\hat{\mu} = M(m)$
    $temp\_list = extract(\Pi, \hat{\mu})$
    $temp\_list = select(temp\_list)$
    $alphabet \leftarrow symb(temp\_list, \hat{\mu})$
  $lift(\Pi, temp\_list)$
```

where $d_{\max}$ is a user defined parameter representing the maximum size allowed for each symbol and *M*(*m*) is the *m*-th metric parameter set in *M*. At the end of the procedure, the symbol list *alphabet* is completely determined. The operator $list_1 = list_2$ for lists assigns to $list_1$ the entire content of $list_2$ destroying the previous content of $list_1$, while the operator $list_1 \leftarrow list_2$ appends to $list_1$ the entire content of $list_2$; the same notation will be used for the same operation on sets. Let's describe the operators *size*, *extract*, *select*, *symb* and *lift* in detail. It is important to remark that in the final *alphabet* exist symbols derived from cluster analyses performed on the basis of different metric parameter sets.

Operator size

The operator *size*(Π) simply returns the size of the subgraphs contained in Π. All the subgraphs in Π have the same number of nodes by construction (see *initialization* and operator *lift*).

Operator extract

The operator $extract(\Pi, \hat{\mu})$ returns a list of clusters of similar subgraphs found in Π. In order to accomplish this task, it performs a hierarchical clustering algorithm on Π, using $\hat{\mu}$ as the metric parameter set. A sequence of *I* partitions is generated. Let's call *J*(*i*) the number of clusters in the *i*-th partition. The partitions are characterized by an increasing number of smaller and more compact clusters. The output of the clustering algorithm we focus on is the entire ensemble of the generated clusters, the number of which is:

$$K = \sum_{i=0}^{I-1} J(i) \; .$$

Each cluster C_k, with $k \in [0, K-1]$, is individually considered as a possible source for a symbol, regardless of the partition in which it is included. Considering both larger (and less compact) and smaller (and more compact) clusters, partially overlapping, introduces a redundancy which improves the robustness of the system. Moreover, as a second output of the clustering algorithm, we store the inclusion tree of the clusters which is naturally known thanks to the hierarchical nature of the procedure, and which is very useful to speed up the inclusion test performed during the step 2 of the operator *select* (see below).

For each cluster C_k, two cost terms $\Phi(C_k)$ and $\Theta(C_k)$ are evaluated. The term Φ is a cardinality cost aiming to reject the scarcely populated clusters; it is defined as:

$$\Phi(C_k) = \begin{cases} 1 - \dfrac{|C_k|}{|S_{tr}|}, \; if \; |C_k| < |S_{tr}| \\ 0, \; otherwise \end{cases} . \tag{3}$$

The term Θ is a compactness cost aiming to obtain clusters constituted by very similar subgraphs; it is defined as:

$$\Theta(C_k) = \frac{\sum_{l=0}^{|C_k|-1} diss_{\hat{\mu}}\left(C_k(l), \tilde{c}_k\right)}{|C_k|}, \tag{4}$$

where $C_k(l)$ is the *l*-th subgraph in the cluster C_k. Firstly both the terms Φ and Θ are normalized between

0 and 1, and then a cost function $F(C_k)$ is evaluated. The function is defined as follows:

$$F(C_k) = (1-\eta)\cdot\Phi(C_k) + \eta\cdot\Theta(C_k), \qquad (5)$$

where $\eta \in [0, 1]$ is a user defined parameter, weighting the two terms.

The K clusters are ordered by increasing values of the cost function F and the final list is returned.

Operator select

The operator *select*(*list*) returns a list containing a subset of the clusters contained in the argument list. In order to obtain the return list, three selection operators are applied in sequence, with the following selection rules:

1. If $F(C_k) \geq \tau_F$, then C_k is deleted. τ_F is a user defined threshold.
2. If $C_h \subseteq C_k$ and $F(C_h) \geq F(C_k)$, then C_h is deleted.
3. If $\Phi(C_k) > \tau_\Phi$, then C_k is deleted. τ_Φ is a user defined threshold.

Operator symb

The operator $symb(list, \hat{\mu})$ generates a symbol for each of the clusters contained in *list*. In order to perform this task, it computes all the data fields constituting a symbol. For all the symbols the size is set equal to the current *size*(Π) and the metric parameter set μ is set equal to $\hat{\mu}$. Moreover, for each different C_k, the centroid $\tilde{c}_k$ and the threshold τ_R are computed. The value of τ_R is defined as follows:

$$\tau_R = (1+\varepsilon_R)\cdot \max_{0\leq l<|C_k|}\left\{diss_{\hat{\mu}}\left(C_k(l), \tilde{c}_k\right)\right\}, \qquad (6)$$

where $\varepsilon_R \geq 0$ is a user defined tolerance.

Operator lift

The operator $lift(\Pi, list)$ clears the search space Π and fills it with subgraphs having a number of nodes increased by one with respect to the previous size. The new subgraphs are chosen from the entire training set in the following way:

```
$\Pi = \varnothing$
for $k = 0$ to $|list| - 1$
  for $l = 0$ to $|C_k| - 1$
    $\Pi \leftarrow extend(C_k(l))$,
```

where $extend(C_k(l))$ is an operator which returns the set of all possible subgraphs obtained adding a new node to $C_k(l)$. The nodes to add are chosen among the ones connected by an edge to at least one of the nodes of $C_k(l)$. Duplicate subgraphs are avoided.

3.2. Symbol recognition (step 2)

During this step, each graph g contained in S_{tr} is mapped into a vector $\mathbf{s} = (s_0, s_1, \ldots, s_{n-1}) \in R^n$, where n is the number of symbols contained in the alphabet. Let's call n_d the number of symbols of size d contained in the alphabet, with $d \in [1, d_{\max}]$. Let's also define by convention $n_0 = 0$. Let's observe that by construction the symbols in alphabet are firstly ordered by increasing sizes. Let Π and C be two auxiliary sets of subgraphs of g, and let $\tau_R(i)$, $\mu(i)$ and $\tilde{c}(i)$ be the τ_R value, the metric parameter set, and the centroid associated with the i-th symbol of alphabet, respectively. For the generic graph $g = (V, E)$ included in S_{tr} the following procedure is applied in order to compute **s**:

```
$\mathbf{s} = (0, 0, \ldots, 0)$
$\Pi = V$
for $d = 1$ to $d_{\max}$
  $C = \varnothing$
  for $i = n_{d-1}$ to $n_d - 1$
    for $l = 0$ to $|\Pi| - 1$
      if $diss_{\mu(i)}(\Pi(l), \tilde{c}(i)) \leq \tau_R(i)$
        $s_i = s_i + 1$
        $C \leftarrow \Pi(l)$
  $temp\_list = \{C\}$
  $lift(\Pi, temp\_list)$
```

As we can observe, at the end of the procedure, the component s_i of the vector **s** contains the number of recognized instances of the i-th symbol of the alphabet in g. The graph g is in fact described by the degree of presence of some basis structures in it. For that reason, we call **s** the s*ymbolic histogram* of g.

4. Tests and classification results

The preprocessed training set $\hat{S}_{tr}$ contains information at a high semantic level, since the symbolic histogram describes the original graphs in terms of presence of significant substructures. Very often, there are multiple combinations of symbolic histogram features containing the information needed for a correct classification of the test patterns. On the other hand, usually, a great number of non-class-discriminating symbol is found, so that the symbolic histogram contains a large number of features which are irrelevant for the classification problem at hand. For that reason it is necessary to adopt a classification system able to perform an automatic selection of the significant input features. The difficulty of the feature selection task depends on the ratio between the number of the class-discriminating symbol combinations and the number of the non-class-discriminating ones. This ratio, in turn, depends on the amount of noise and distortion in the original graphs.

In order to evaluate the performances of the symbolic analysis system described above, we fed a *K–nn* classifier with the preprocessed training set $\hat{S}_{tr}$. The *K* value is a user defined parameter. The selection of relevant features is heuristically performed by a genetic algorithm led by the minimization of a cost function based on the Occam's Razor Criterion:

$$\Omega = (1-\lambda)\cdot E_{tr} + \lambda \cdot \Gamma\,, \tag{7}$$

where E_{tr} is the classification error percentage on $\hat{S}_{tr}$ and Γ is a measure of the structural complexity defined as the normalized number of the employed input features: $\Gamma = \frac{|selected\ features|}{n}$. The weight $\lambda \in [0,1]$ is a user defined parameter. The *K–nn* model is used to classify the patterns in $\hat{S}_{ts}$. Before being submitted to the classification system, the patterns in both $\hat{S}_{tr}$ and $\hat{S}_{ts}$ are normalized between 0 and 1.

In order to test the whole classification system, synthetic data have been used. The node and edge labels of the algorithmically generated graphs mimic the feature vectors describing the discrete objects detected in an image by a segmentation algorithm (as in [1]), together with their spatial relationships. The node labels belong to R^6 and they are sectioned as follows: $\mathbf{v} = (\mathbf{v}_0;\mathbf{v}_1;\mathbf{v}_2)$, where $\mathbf{v}_0 = (x,y)$ describes the position of the object, $\mathbf{v}_1 = (r,g,b)$ describes the color of the object and $\mathbf{v}_2 = (s)$ describes the size of the object. The edge labels belong to R^2 and they are constituted by only one section $\mathbf{e} = (\mathbf{e}_0)$, where $\boldsymbol{e}_0 = (\delta x, \delta y)$ is the difference vector between the positions of the connected objects. Each feature is normalized between 0 and 1. Thanks to this definition of node and edge labels, the synthetic graphs and subgraphs, and even their computed centroids, can be displayed in the form of a digital image on the PC screen. This allows to get a useful visual representation of the patterns and of the computed symbol centroids, which helps to understand the behavior of the system.

The algorithm which generates the patterns receives as input a *problem definition*, structured as follows. Let $G = \{g_0, g_1, \ldots, g_{n-1}\}$ be a set of *n* graphs which are the significant substructures constituting the basis for the class definitions. Let's call a *pattern template* a set of pairs of the type $< g_j, r >$, where $j \in [0, n-1]$ and *r* is a positive integer indicating the number of instances of that graph in the synthetic pattern to be generated. Let $T = \{t_0, t_1, \ldots, t_{m-1}\}$ be a set of *m* pattern templates and $Q = \{Q_0, Q_1, \ldots, Q_{k-1}\}$ be a set of *k class descriptors*, where each of them is a subset of *T*. When generating a pattern of a given class, one of the templates is randomly selected and used to determine the pattern structure. As an example, if a class is represented by the subset $\{t_0, t_2\}$ this means that a pattern belonging to that class can correspond to the template t_0 XOR t_2. In order to have a consistent definition of the classification problem, different sets included in *Q* must have a void intersection, i.e. $Q_p \cap Q_q = \varnothing$ for $p \neq q$. A problem definition is constituted by a triplet $< G,T,Q >$. In order to generate a graph pattern $\hat{g}$ belonging to the *p*-th class, the algorithm works as follows:

1. Initialize $\hat{g}$ as empty.
2. Randomly select one value from Q_p, determining which of the m pattern templates has to be used; let's call it $\hat{t}$.
3. For each pair $< g_j, r >$ included in $\hat{t}$, insert in $\hat{g}$ the nodes of g_j, adding to the $\mathbf{v}_0$ section of each node the same randomly generated translation vector; repeat the operation *r* times, each time with a new translation vector.
4. Alter each feature of all the nodes added so far to $\hat{g}$ by a noise contribute extracted from a normal distribution $N(0, \sigma_N^2)$, where σ_N^2 is a user defined parameter.
5. Add a random number of randomly generated nodes. The number is chosen from a discrete uniform distribution in the interval $[0, N_{\max}]$, where $N_{\max}$ is a user defined parameter.
6. For each couple of nodes, compute the spatial relationship and add the corresponding edge. This

way, $\hat{g}$ is a complete graph, and this represents the worst case for the complexity of the symbol search procedure.

TABLE I
PROBLEM DEFINITIONS

Code	$\|G\|$	$\|\gamma\|$	*Size*	$\|T\|$	$\|Q\|$
D1	4	4	2, 3	4	4
D2	2	2	3	2	2
D3	6	6	3	6	2
D4	10	2	3	10	2
D5	20	2	3	20	2
D6	2	2	7	2	2

Details about the six problem definitions.

Six different problem definitions have been created to test the system. In Table I some details about the definitions are given. As it can be observed, the number of elements in *G* doesn't necessarily coincides with the number of actual, "logical" substructures conceived to define the problem. In fact, different elements in *G* could be different determinations of the same logical substructure. For example, if a substructure is characterized by size and spatial relationships of the objects, the color of the objects is irrelevant; thus different elements in *G* could represent differently colored versions of the same substructure. For that reason the table I shows a column labeled with $|\gamma|$ representing the number of actual significant substructures. Obviously, we always have $|\gamma| \le |G|$. The column "Size" shows the number of nodes of the significant substructures. For each problem definition four different instances have been generated, with increasing values of σ_N^2 and $N_{\max}$; table II shows the salient characteristics of the resulting 24 problem instances.

For each classification problem instance, the symbolic analysis has been performed, and the corresponding $\hat{S}_{tr}$ and $\hat{S}_{ts}$ have been used to train and test the *K–nn* classifier. Due to the randomness of the feature selection heuristic, the employed genetic algorithm has been run 50 times for each instance, yielding in general different classification error percentages on $\hat{S}_{ts}$. In order to try out the automation degree of the system, all the 24 instances have been faced with a unique setting of the user parameters; in this regard, we have fixed them to the following values: $\eta = 0.9$, $\tau_F = 0.15$, $\tau_\Phi = 0.8$, $\varepsilon_R = 0.1$, $d_{\max} = 3$, $K = 5$, $\lambda = 0.25$.

TABLE II
PROBLEM INSTANCES AND CLASSIFICATION RESULTS

(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)
1	D1	80	0	0	0	0.75	1.03
2	D2	40	0	0	0	0	0
3	D3	120	0	0	0	0	0
4	D4	40	0	0	0	0	0
5	D5	40	0	0	0	1.2	2.14
6	D6	40	0	0	0	0	0
7	D1	80	10^{-4}	3	0	0.6	1.86
8	D2	40	10^{-4}	3	0	1.45	2.07
9	D3	120	10^{-4}	3	0	2.12	2.38
10	D4	40	10^{-4}	3	0	1.8	1.94
11	D5	40	10^{-4}	3	0	0	0
12	D6	40	10^{-4}	3	0	1.4	2.46
13	D1	80	$3 \cdot 10^{-4}$	4	0	2.93	7.61
14	D2	40	$3 \cdot 10^{-4}$	4	0	7.2	6.51
15	D3	120	$3 \cdot 10^{-4}$	4	0	2.98	3.21
16	D4	40	$3 \cdot 10^{-4}$	4	0	1.95	1.52
17	D5	40	$3 \cdot 10^{-4}$	4	0	8.15	6.55
18	D6	40	$3 \cdot 10^{-4}$	4	0	2.6	3.74
19	D1	80	$5 \cdot 10^{-4}$	5	0	3.7	9.9
20	D2	40	$5 \cdot 10^{-4}$	5	0	15.35	8.6
21	D3	120	$5 \cdot 10^{-4}$	5	1.67	4.97	5.42
22	D4	40	$5 \cdot 10^{-4}$	5	0	1.95	3.09
23	D5	40	$5 \cdot 10^{-4}$	5	2.5	12.8	4.99
24	D6	40	$5 \cdot 10^{-4}$	5	0	2.05	2.81

Details about the problem instances and classification results. The columns represent the following characteristics: (a) problem instance number, (b) original problem definition, (c) number of patterns in training and test set, (d) σ_N^2 value, (e) $N_{\max}$ value, (f) minimum error percentage on test set, (g) mean error percentage on test set, (h) standard deviation of error percentage on test set. The values in columns (f-h) have been computed over a batch of 50 runs of the genetic algorithm.

As shown in Table II, the minimum error percentage on the test set, achieved for each instance during the 50 runs of the genetic algorithm, is always 0% except for two cases (instances 21 and 23). The average error exceeds 10% in only two cases (instances 20 and 23). If the instances are grouped by amount of noise it can be seen that, as a general trend, both mean and standard deviation of the error increase with noise, as expected.

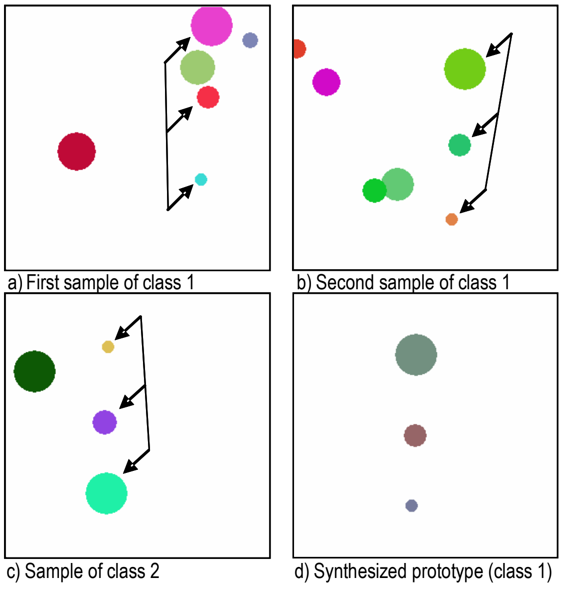
a) First sample of class 1
b) Second sample of class 1
c) Sample of class 2
d) Synthesized prototype (class 1)

Figure 1. The figure shows some sample patterns (a-c) from the training set of the problem instance 22, with linked arrows indicating the relevant substructures, and the centroid (d) of the symbol selected by the genetic algorithm to discriminate the classes. The symbol corresponds to the relevant substructure found in class 1 and has an associated metric parameter set which gives a non-null weight to the sections $\mathbf{v}_2$ and $\mathbf{e}_0$, as it should be, since the color and absolute position of the nodes composing the substructure doesn't characterize the symbol and can assume different (random) values throughout the whole set of class examples. We can observe how all the node and edge features have been averaged to form the prototype.

These results confirm that the useful information contained in $\hat{S}_{tr}$ and $\hat{S}_{ts}$ becomes harder to extract for the heuristic search engine, as noise increases. Thus, more complex and accurate search mechanisms could improve the overall performance. All the tests have been carried out on a PC equipped with AMD Athlon™ 64 2800+ 1.81 GHz CPU, 1 GB of RAM; software has been developed with the Borland C++ Builder 6™ compiler under the Windows XP™ operative system. The time required for the symbols discovery procedure ranges from 16'' to 35' 1'' depending on the problem instance, with an average of 10' 33''. The time required for the symbol recognition on a single pattern ranges from 0.001'' to 5.831'' with an average of 0.708''. The time required for a single run of the genetic search algorithm, including the cost evaluation of the candidate models, ranges from 0.292'' to 31.833'', with an average of 7.587''.

As it can be observed, good performances are obtained even in the instances derived from the problem definition D6, which is based on substructures of size equal to 7. In fact, even if $d_{\max} = 3$, the problem can be solved because class-discriminating, smaller subgraphs of the substructures can be isolated. However, as concerns the semantic information extraction, the solution can be considered incomplete because the synthesized prototypes represent just subgraphs of the real significant structures.

5. Conclusions and future work

In this paper we propose a symbolic classification system based on an information granulation procedure, able to deal with graphs with node and edge labels belonging to continuous vector spaces. The overall classification technique is general enough to be employed on a wide range of problems coming from real world applications. It can be thought as an essential step to define a unified granular modeling approach. The system is able to solve a large set of different classification problems automatically, i.e. without any need to set up system parameters on the basis of the problem at hand. In general, our approach showed a good aptitude to robustness, since even noisy problems have been solved without any need to adapt system parameters to the noise level. The extraction of relevant substructure prototypes, each characterized by its own local metric parameters, appeared to be a powerful framework for the synthesis of alternative representations of the original graphs. In fact the symbolic histogram is a high semantic level representation that can be used to feed directly classification systems working on metric spaces.

6. References

[1] A. Rizzi, G. Del Vescovo, "Automatic Image Classification by a Granular Computing Approach" in *Proc. IEEE International Workshop on Machine Learning for Signal Processing*, Maynooth, Ireland, 2006, pp. 33 - 38.

[2] M. Deshpande, M. Kuramochi, N. Wale, G. Karypis, "Frequent substructure-based approaches for classifying chemical compounds" *IEEE Trans. Knowledge and Data Engineering*, vol. 17, no. 8, pp. 1036–1050, August. 2005.

[3] L. Holder, D. Cook, J. Gonzales, I. Jonyer, "Structural Pattern Recognition in Graphs" in D. Chen, X. Chengs (Eds.) *Pattern Recognition and String matching*, Series: Combinatorial Optimization, Vol. 13, ISBN: 978-1-4020-0953-2, Kluwer Academic Publishers, pp. 255-279, 2003.

[4] X. Jiang, A. Münger, H. Bunke, "On Median Graphs: Properties, Algorithms, and Applications", in *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 23, No. 10, October 2001, pp. 1144-1151.