# Subgraph Isomorphism Detection with Support for Continuous Labels

**Gerardo Perez** [1], **Ivan Olmos** [1] and **Jesus A. Gonzalez** [2]
[1] Facultad Ciencias de la Computacion
Benemerita Universidad Autonoma de Puebla
14 sur y Av. San Claudio, Ciudad Universitaria, Puebla, Mexico
{onofre.gerardo, ivanoprkl}@gmail.com
[2] Instituto Nacional de Astrofisica, Optica y Electronica,
Luis Enrique Erro No. 1, Sta. Maria Tonantzintla, Puebla, Mexico
jagonzalez@inaoep.mx

## Abstract

This paper presents an algorithm for the subgraph isomorphism detection with support for continuous and discrete labels. This algorithm consists of three phases: first, we perform the discretization of continuous labels through unsupervised discretization methods; in the second step, the SI-COBRA algorithm is used for the subgraph isomorphism detection; finally, in the third phase, we need a simple transformation in order to show the results with a logical meaning with respect to the original data.

## Introduction

A graph-based representation is a useful technique used in different areas, such as data mining and machine learning, which work with structured domains in which we can not represent their features as a set of attributes. Some examples of these domains are the links between web pages, computer network topologies, electrical circuits, and chemical structures, among others (Cook, Holder, and Djoko 1995).

For example, in Fig. 1a we can see the electrical circuit features domain representation using a labeled graph, which keeps both structure and the necessary information for its analysis. In this case is difficult to represent the circuit through a record from a relational database. Similarly, the chemical structure in Fig. 1b has a natural representation through graphs.

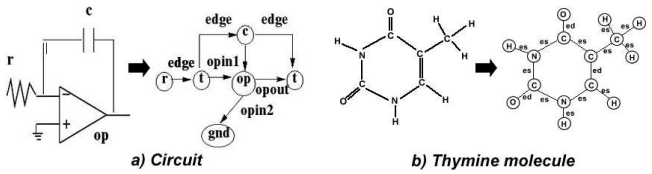



Figure 1: Structured Domains

When we use this type of representation, one of the most important tasks is to find common structures in a graph, or set of graphs, in order to discover interesting behaviors of the data. Another similar task is counting and detecting the number of instances of a specific graph in a set of graphs. These tasks are based on finding similarities between graphs/subgraphs, which is known as Graph Matching (Cook, Holder, and Djoko 1995).

There are different ways of interpreting the information represented by the labels of the vertices and edges. In our approach, the labels of the edges represent an attribute name or a relationship between a pair of vertices. So, this type of label will be associate with discrete values, for example ”weight”, ”high” or ”molecular link”, according to the represented feature. On the other hand, the labels of the vertices represent the possible values that the attribute or relationship can take and for that reason this type of label may contain both, discrete and continuous values.

There are several algorithms that work with labeled graph - based representations, such as Subdue (Cook and Holder 1994), AGM (Inokuchi, Washio, and Motoda 2000), FSG (Kuramochi and Karypis 2002), gSpan (Yan and Han. 2002), SEuS (Ghazizadeh and Chawathe 2002), FSP(Han and Kamber 2006), Mofa(Borgelt and Berthold 2002), Edgar(Wörlein et al. 2006), Gaston(Nijssen and Kok 2004), which deal with the Data Mining problem and others such as SI-COBRA (Olmos, Gonzalez, and Osorio 2005), which focus on the subgraph isomorphism detection problem.

However, the graph-based algorithms described before consider that the values associated with the labels of the vertices come from discrete attributes. Nevertheless, there are several domains that have both discrete and continuous features. Therefore, we can not work with such domains in an appropriate way.

In Fig. 2a and 2b we can notice that all the numeric values are different and for this reason there were found no valid associations (matches) for this example. However, although there is no accurate association, we can observe that there are vertices with values (represented by the labels of the vertices) that are numbers very close to each other, so we could consider that both values are within the same range and we could consider these labels as a valid match.

On the other hand, by using a discretization approach, as we show in figures 2c and 2d, which maps the set of data to a set of labels represented by discrete intervals and perform a search for similar structures, it is possible to find valid associations (matches) for the same example.

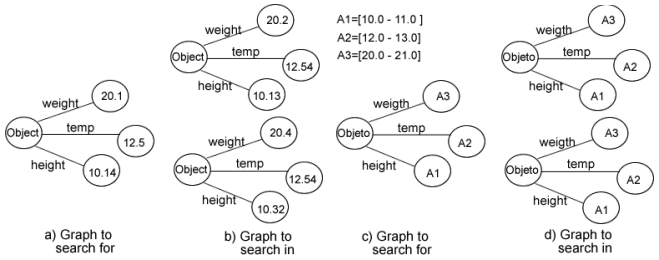



Figure 2: Example

Based on the above mentioned, we propose a SI-COBRA extension called SI-COBRA-C (continuous) algorithm, which supports both continuous and discrete values in vertices' labels and performs the features discretization through unsupervised discretization methods.

## Discretization Methods

The aim of the discretization step is to build a smaller set of intervals (bins) in which numerical values can be brought together, trying to identify a relationship among instances of the same interval. In other words, the problem of discretization consists of defining 2 things: 1) the number of intervals (k) and 2) bounds (cut-points) for each interval generated.

There are different axes to classify the discretization methods (Dougherty, Kohavi, and Sahami 1995) such as global, local, supervised, unsupervised, static, and dynamic. Since the goal of our approach is to perform an independent discretization process for each continuous feature, whether or not an class feature exists, we are interested in unsupervised methods because these do not require class information.

There are 2 well-known unsupervised discretization methods called EqualWidth and EqualFrequency which are easy to implement. These methods divide the range of observed values into k intervals of equal size or into k intervals so that each interval contains approximately the same number of instances, respectively. Nevertheless, these simple methods do not take into account the dispersion of data.

On the other hand, there are clustering algorithms that divide the data into groups of similar objects such as k-means or EM (Han and Kamber 2006), which calculate the centroids of the clusters so that these clusters will form intervals, whose boundaries are determined by the midpoint between 2 centroids. In clustering methods, we can use different similarity measures to assign the data to clusters such as the Euclidean, Manhattan, and Minkowski distances, etc. (Han and Kamber 2006).

Other techniques are based on density measurements such as Kernel Density Estimator (Biba et al. 2007) and TUBE (Schmidberger and Frank 2005), which in a binary way cut the data set looking for the best cut-points to generate a tree in which each leaf represents an interval. The advantage of these methods is that they try to adjust to the real data distribution, however for some distributions it can generate a high number of intervals.

All the strategies mentioned above, require a number k to know how many intervals (maximum or exact) should be generated. In some of these methods that number is defined by the user. However, one of our purposes is to calculate the number of intervals automatically in a process transparent to the user. In this sense, different approaches have been proposed (see Figure 3) to determine the value of k for a data set. The simplest one, known as Ten bins creates 10 intervals regardless of the distribution, frequency and dispersion of data. Some authors (Dougherty, Kohavi, and Sahami 1995) (Biba et al. 2007) (Yang and Webb 2002) (Agre and Peev 2002) have based their work on this value using EqualWidth and / or EqualFrequency, though not all of them support the selection of this number of intervals.

Other approaches use rules to define the value of k. For example, Dougherty (Fig. 3b) (Boulle 2005) and Sturges (Fig. 3c) (Sturges 1926) use the logarithm of the number of different data values to define the value of k. On the other hand, Scott (Fig. 3d) (Scoot 1979) and Freedman-Diaconis (Fig. 3e) (Freedman and Diaconis 1981) calculate the bin width (h) based on the standard deviation and interquartile range respectively and they define $k$ based on the calculated width. For these reasons, these strategies can be applied to the EqualWidth method.

There are other methods that use more complex rules taking into account the data dispersion and minimizing or maximizing the gain generated by the creation of one interval, an example of these methods are Birgé (Fig. 3f) (Birgé and Rozenholc 2002) and cross-validated log-likelihood (Fig. 3g) (Biba et al. 2007), (Schmidberger and Frank 2005). The Birgé's formula can be used in EqualWidth because it generates equal sized intervals. On the other hand, the cross-validated loglikehood method can be used in algorithms based on density, EqualWidth (Biba et al. 2007) (Schmidberger and Frank 2005) and clustering algorithms.

The methods described above do not always produce an appropriate partition scheme for a given data set. Therefore, we can not choose only one method of discretization for any dataset. Based on the previous, if we require to generate a number of partitions for any set of arbitrary data, we need to evaluate different strategies in order to choose the scheme that best fits the characteristics of the data.

## Notation

In this section, we will define some of the concepts used along the document.

Let $G$ be a labeled graph, let $AT$ be the set of different attributes in $G$ and let $\aleph^C$ and $\aleph^D$ be the number of continuous and discrete features represented by $G$. $AT$ is defined as $AT = \{A^C \bigcup A^D\}$, such that $A^C = \{A^C_k | k = 1, \ldots, \aleph^C\}$ is the continuous features subset of $G$ and $A^D = \{A^D_k | k = 1, \ldots, \aleph^D\}$ is the discrete features subset of $G$. $D(A^C_k)$ represents the domain of $A^C_k$. We also define $\gamma$ as the function used to obtain the attribute name for a vertex in $A^C$.

As a result, a graph $G$ will be a 6 - tuple $G =$

| |
|---|
| $n$ = number of different values<br>$x$ = set data |
| a) Ten bins k=10 |
| b) Dougherty k = max (1,2*log($n$)) |
| c) Sturges k = 1 + log($n$) |
| d) Scott h = $\frac{3.5\sigma}{n^{1/3}}$, $\sigma$ = standar desviation, k=$\frac{max(x)-min(x)}{h}$ |
| e) Freedman-Diaconis h = 2*$\frac{IQR(x)}{n^{1/3}}$, IQR=Interquartile range |
| f) Birgé. Consists of maximizing $L_n(I) - pen(I)$ for $1 \leq I \leq n/log(n)$ where $L_n(I) = \sum_{i=1}^{I} n_i log(n_i I/n)$ is the loglikelihood of the histogram with I bins and $pen(I) = I$-1+$log(I)^{2.5}$ is the penalty |
| g) Cross-validated loglikelihood (LL).<br>LL = $\sum_{i=1} n_{i_{test}} log(\frac{n_{i_{train}}}{w_i * N})$ where $n_{i_{test}}$= test instances that fall in $bin_i$, $n_{i_{train}}$= train instances that fall in $bin_i$, N= total instances and $w_i$= bin width. |

Figure 3: Strategies to define the number of intervals

$(V, E, L_V, L_E, \alpha, \beta)$, where:

- $V = \bigcup_{k=1,\ldots,\aleph^C} V_k \bigcup V_{DISC}$,$V \neq \emptyset$, where $V_k$ is the finite set of vertices coming from $A_k^C$ and $V_{DISC}$ is the finite set of vertices coming from $A^D$.
- $E \subseteq V \times V$, is the finite set of edges, $E = \bigcup_{k=1,\ldots,\aleph^C} E_k \bigcup E_{DISC}$, $E_k = \{e = \{v_i, v_j\} : v_i \in V_k$ or $v_j \in V_k\}$, $E_{DISC} = \{e = \{v_i, v_j\} : v_i, v_j \in V_{DISC}\}$.
- $L_V = \bigcup_{k=1,\ldots,\aleph^C} L_V^k \bigcup L_V^{DISC}$, where $L_V^k = D(A_k^C)$ is the finite set of labels of vertices coming from $V_k$, $L_V^{CONT} = \{L_V^k | k = 1, \ldots, \aleph^C\}$, and $L_V^{DISC}$ is a finite set of vertices that comes from $V_{DISC}$.
- $L_E = L_E^{CONT} \bigcup L_E^{DISC}$, is a finite set of labels of edges where $L_E^{CONT} = \{A_k^C : \gamma(L_V^k) = A_k^C, \forall k = 1, \ldots, \aleph^C\}$, $L_E^{DISC}$ = set of labels of edges from $A^D$.
- $\alpha = \bigcup_{k=1,\ldots,\aleph^C} \alpha_k : V_k \rightarrow L_V^k \bigcup \alpha_{DISC} : V_{DISC} \rightarrow L_V^{DISC}$, where $\alpha_k$ is a function assigning labels to vertices in $V_k$ and $\alpha_{DISC}$ is the function assigning labels to vertices in $V_{DISC}$.
- $\beta = \bigcup_{k=1,\ldots,\aleph^C} \beta_k : E_k \rightarrow L_E^{CONT} \bigcup \beta_{DISC} : E_{DISC} \rightarrow L_E^{DISC}$, where $\beta_k$ is a function assigning labels to edges in $E_k$ and $\beta_{DISC}$ is a function assigning labels to edges in $E_{DISC}$.

A subgraph $G^S$ of $G$, denoted by $G^S \subseteq G, G^S = (V^S, E^S, L_V^S, L_E^S, \alpha^S, \beta^S)$ is a graph such that $V^S \subseteq V, E^S \subseteq E, L_V^S \subseteq L_V, L_E^S \subseteq L_E, \alpha^S \subseteq \alpha$ and $\beta^S \subseteq \beta$.

Given 2 graphs $G' = (V', E', L_V', L_E', \alpha', \beta')$ and $G = (V, E, L_V, L_E, \alpha, \beta)$, we say that $G'$ is isomorphic to $G$, denoted by $G' \cong G$, if there are bijections $f : V' \rightarrow V$ and $g : E' \rightarrow E$, where:

- $\forall v' \in V', \alpha'(v') = \alpha(f(v'))$
- $\forall \{v_i', v_j'\} \in E', \beta'(\{v_i', v_j'\}) = \beta(g(\{v_i', v_j'\}))$

Let $G'$ and $G$ be two graphs, where $G'$ is a subgraph isomorphic to $G$ if there exists any $G^S \subseteq G$ such that $G' \cong G^S$.

For example, let $G = (V, E, L_V, L_E, \alpha, \beta)$ be the graph shown in Fig. 4. We can notice two types of features or attributes according to the labels values associated with the vertices, so that:

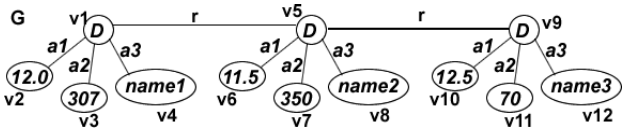



Figure 4: Graph $G$

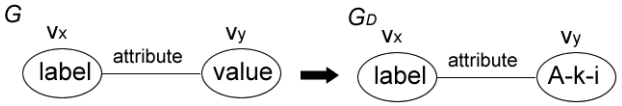
- $AT = \{A_1^C, A_2^C, A_1^D, A_2^D\}$, $A_1^C = a_1$, $A_2^C = a_2$ because the labels of the features associated with the vertices $a_1$ and $a_2$ are (continuous) numerical values and $A_1^D = a_3$ because the labels of the vertices associated with feature $a_3$ are (discrete) characters.
- $V = \{v_1, v_2, \ldots, v_{12}\}$, $V_1 = \{v_2, v_6, v_{10}\}$, $V_2 = \{v_3, v_7, v_{11}\}$ and $V_{DISC} = \{v_1, v_4, v_5, v_8, v_9, v_{12}\}$.
- $E$ = $\{\{v_1, v_2\}, \{v_1, v_3\}, ..., \{v_9, v_{12}\}\}$,
  $E_1$ = $\{\{v_1, v_2\}, \{v_5, v_6\}, \{v_9, v_{10}\}\}$,
  $E_2$ = $\{\{v_1, v_3\}, \{v_5, v_7\}, ..., \{v_9, v_{11}\}\}$,
  $E_{DISC}$ = $\{\{v_1, v_4\}, \{v_1, v_5\}, \{v_5, v_8\}, \{v_5, v_9\}, \{v_9, v_{12}\}\}$
- $L_V$ = {12.0, 307, name1, 11.5, 350, name2, . . . ,12.5 ,70, nameN} such that $L_V^{CONT} = \{L_V^1, L_V^2\}$, $L_V^1$ = {12.0, 11.5 , . . .12.5}, $L_V^2$ = {307, 350, . . . ,70}, $L_V^{DISC}$ = {D, name1, . . . , nameN}
- $L_E = \{a_1, a_2, a_3, r\}$, $L_E^{CONT} = \{a_1, a_2\}$, $L_E^{DISC} = \{a_3, r\}$
- $\alpha_1$ ($V_2$) = 12.0, $\alpha_2$ ($V_3$) = 30.7, $\alpha_1$ ($V_6$) = 11.5, $\alpha_2$ ($V_7$) = 350, $\alpha_1$ ($V_{10}$) = 12.5, $\alpha_3$ ($V_{11}$) = 70, $\alpha_{DISC}$ ($V_1$) = $D$, $\alpha_{DISC}$ ($V_4$) = $name1$
- $\beta_1$ ( $\{V_1, V_2\}$) = $a1$, $\beta_1$ ( $\{V_5, V_6\}$) = $a1$, $\beta_2$ ( $\{V_1, V_3\}$) = $a2$, $\beta_2$ ( $\{V_9, V_{11}\}$) = $a2$, $\beta_{DISC}$ ( $\{V_1, V_4\}$) = $a3$.

## The SI-COBRA-C method

With the aim to solve the subgraph isomorphism detection in domains with continuous features, we propose the following algorithm that implements a strategy to deal with this type of labels through a discretization process.

Before describing how the algorithm works, it is necessary to define $\gamma$ such that $\gamma : L_V^{CONT} \rightarrow A^C$, which is an auxiliary function to obtain the name of the feature associated with the vertices in set $V_k$. In the example shown in Fig. 4, $\gamma(L_V^1) = A_1^C$, $\gamma(L_V^2) = A_2^C$.

Algorithm 1 shows the sequence of steps to follow in order to identify isomorphic subgraphs able to deal with numerical features. The algorithm first creates the set $\hat{L}_V$ containing the sets of all possible values for each of the features in $G$ and $G'$ such that $G$ is the graph to search in and $G'$ is the graph to search for.




If attribute $\in A_k^C$ and $v_y \in V_k$, then for $GD$ $\alpha(v_y) \leftarrow$ A-k-i such that value > A-k-i.lower and value < A-k-i.upper

Figure 5: Example of assigning discrete labels

After that, in line 2 of algorithm 1, each of the sets in $\hat{L}_V$ is discretized creating the set $\hat{L}_{VD}$ containing all the labels for each feature generated by the discretization process. This set is joined with the discrete labels set in order to have a unique set as defined in line 3 of the same algorithm.

Once we created the set, we define (in lines 4 and 5) 2 new graphs $G_D$ and $G'_D$ that will be the discretized versions of the input graphs. Each label represents an discrete interval and it has two values associated which represent the lower and upper range. So, the discrete label A-k-i represents the i-th interval for the k-th feature, such that A-k-i.lower and A-k-i.upper represent the lower and upper bounds of the interval, respectively. In Figure 5, we show that the graph $G_D$ maintains the same vertices and edges as $G$ and only the continuous labels of the vertices are modified.

The discretized graphs $G_D$ and $G'_D$ will be the input graphs to the SI-COBRA algorithm. SI-COBRA generates $Gset$ containing the set of detected isomorphic graphs. The labels detected in isomorphic graphs represent discrete intervals, for which we need to add a process to represent the data with the original domain feature values (last line of algorithm 1).

It is important to mention that the last phase is easy to implement because there is a direct relation between the vertices of $G_D$ and $G$, and therefore using the function $\alpha$ we can recover the original values of the vertices.

**Algorithm 1** SI-COBRA-C

**Input:** $G$,$G'$
**Output:** Isomorphic subgraphs found
1: $\hat{L}_V \leftarrow \{L_V^k \bigcup L_V'^j | \gamma(L_V^k) = \gamma'(L_V'^j)\} \forall k = 1,\ldots,\aleph^C, j = 1,\ldots,\aleph'^C$
2: $\hat{L}_{VD} \leftarrow Discretization(\hat{L}_V)$
3: $L_{VD} \leftarrow \hat{L}_{VD} \bigcup L_V^{DISC}$
4: $G_D \leftarrow (V, E, L_{VD}, L_E, \alpha, \beta)$
5: $G'_D \leftarrow (V', E', L_{VD}, L'_E, \alpha', \beta')$
6: $G_{set} \leftarrow SI\text{-}COBRA(G_D, G'_D)$
7: $Results(G_{set}, G)$

Algorithm 2 works with the data set included in $\hat{L}_V$. Each of these sets is discretized independently of the other sets, so for each set we want to find the method that best fits the data. Once we generated discrete labels in line 4 of Algorithm 2 we create a unique set which is returned to algorithm 1.

Since we can not choose only one method that works well for all datasets, our approach is to test different methods and then perform a comparison of the results generated in order to select the method that best fits the data.

**Algorithm 2** Discretization

**Input:** $\hat{L}_V$
**Output:** $\hat{L}_{VD}$ set of discretized labels
1: **for** $k = 1$ to $|\hat{L}_V|$ **do**
2: $\hat{L}_{VD}^k \leftarrow Disc\_method(L_V^k)$
3: **end for**
4: $\hat{L}_{VD} = \bigcup_{k=1,\ldots,|\hat{L}_{VD}|} \hat{L}_{VD}^k$
5: return $\hat{L}_{VD}$

**Algorithm 3** $Disc\_method$

**Input:** $\hat{L}_V^k$
**Output:** $\hat{L}_{VD}^{temp}$ set of discrete labels
1: cutPoints vector of cut-points' vectors
2: $\hat{L}_{VD}^{temp} \leftarrow \varnothing$
3: cutPoints[1] $\leftarrow$ EqualWidth($\hat{L}_V^k$)
4: cutPoints[2] $\leftarrow$ EqualFrequency($\hat{L}_V^k$)
5: cutPoints[3] $\leftarrow$ Clustering($\hat{L}_V^k$)
6: cutPoints[4] $\leftarrow$ Density($\hat{L}_V^k$)
7: **for** $i$ = 1 to |cutPoints| **do**
8: Error[i] $\leftarrow$ Evaluation($L_V^k$,cutPoints[i])
9: **end for**
10: l $\leftarrow$ i | Error[i] = min(Error[i]) $\forall$ i=1,. . .,|Error|
11: **for** $i$=1 to | cutPoints[l]|-1 **do**
12: A-k-i.lower=cutPoints[l][i]
13: A-k-i.upper=cutPoints[l][i+1]
14: $L_{VD}^{temp} \leftarrow L_{VD}^{temp} \bigcup$ {A-k-i}
15: **end for**
16: return $\hat{L}_{VD}$

Therefore, in lines 3-6 of algorithm 3 the data set is discretized by different methods. We will test the data set with classical methods such as EqualWidth and EqualFrequency, with a clustering method and a with a method based on density. Each of these methods generates a vector containing the cut-points with which we can define the limits of the intervals. In lines 7-9 of algorithm 3 we evaluate the intervals generated by each method through the $Evaluation$ function.

After selecting the method with the lowest error, the labels are generated to represent discrete intervals. The function $Evaluation$ measures the variance of each of the intervals generated and the average sum of these variances define the $error$ associated with the scheme evaluated. By using the variance, we look for the scheme in which the data is less dispersed within the ranges. Therefore $error = \frac{1}{k}\sum_{i=1}^{k} var(k)$ where $k$ = number of intervals, $var(k)$ = variance of data that falls in the interval k.

## Example

We show the following experiment performed with a simple database called auto-mpg downloaded for free from the following website http://www.ics.uci.edu/ mLearn/MLRepository.html.

It is important to mention that the graph-based representation is not restricted to structured domains, because there is possible to find an appropriate representation according to the analyzed domain.

The auto-mpg database is formed by 411 different data examples. The data base has 9 features, where 6 were selected to perform the experiments. The selected features were a1 = mpg (continuous), a2 = displacement (continuous), a3 = horsepower (continuous), a4 = weight (continuous), a5 = acceleration (continuous) and a6 = name (discrete), which are shown in Fig. 6. To represent the data using a graph-based representation we chose a star topology as shown in Fig. 7 in which each central node represents an entry in the database and the surrounding vertices represent the different features.

| | a1 | a2 | a3 | a4 | a5 | a6 |
|---|---|---|---|---|---|---|
| 1 | 18 | 307 | 130 | 3504 | 12 | "malibu" |
| 2 | 15 | 350 | 165 | 3693 | 11.5 | "buick_320" |
| 410 | 32.7 | 168 | 132 | 2910 | 11.4 | "datsun_280-zx" |
| 411 | 23.7 | 70 | 100 | 2420 | 12.5 | "mazda" |

Figure 6: auto-mpg data

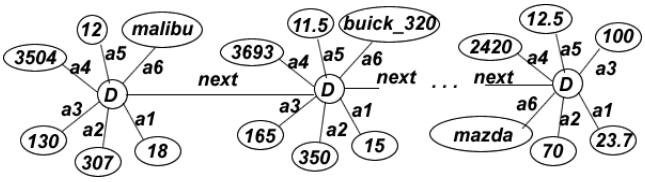



Figure 7: Graph-based representation. Graph $G$

In this experiment we used 2 methods to perform the discretization of each feature in order to observe the number of intervals generated by each of them. The methods used were EqualWidth using Scott's rule and Clustering using Euclidean distance and EM with 10 iterations.

In our experiments, we want to identify all subgraphs with the following structures: Graph 1 {a1=16.4, a2=306, a5=12.1} and Graph 2 {a3=130, a4=3500}. In Fig. 8, we can observe the representation of the graphs to search for.

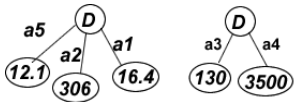



Figure 8: Graphs to search for. Graphs G1 and G2 respectively

In first place we need to discretize the input graphs in order to generate their discrete versions, which will be the input to the SI-COBRA algorithm.

In Fig. 8 we can see the graph-based representation for the graphs to search for. The outcomes found are shown in Fig. 9, in which the first column shows the number of associations found using the input graphs directly, ie. without using the discretization process. As we can see, there are no valid results because all the values are different. In the second and third column, we show the identified valid associations using clustering and EqualWidth respectively. We can observe that using a discretization process we can find matches in which the original values are close to each other

| | Normal | Clustering | EqualWidth |
|---|---|---|---|
| Graph 1 | 0 | 14 | 19 |
| Graph 2 | 0 | 12 | 5 |

Figure 9: Results

In the figure 10, we show the discretized graph generated by the clustering and EqualWidth methods, respectively.

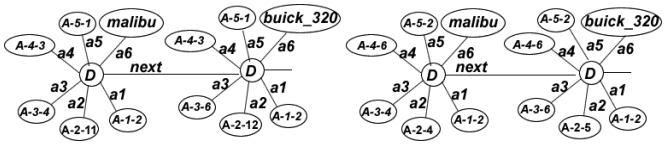



Figure 10: Discretized graphs

By searching on discretized graphs it is possible to identify valid matches, because we are not looking for a specific value, we are looking for the interval in which the searched value falls.

Once we found the isomorphic subgraphs, we recover the original values of the vertices. In Fig. 11, we show the values reported as isomorphism subgraph. The table shows the vertices values that come from the detected isomophism subgraph to $G2$ using EqualWidth and Clustering, respectively. The shaded line remarks the values of $G2$ to search for. Although there does not exist a datum that meets the desired feature, we can see that there are very approximate data values (with respect to the dimension of the data range).

a)

| | a4 | a3 |
|---|---|---|
| R1 | 3725 | 129 |
| R2 | 3605 | 125 |
| R3 | 3504 | 130 |
| G 2 | 3500 | 130 |
| R4 | 3410 | 120 |
| R5 | 3410 | 133 |

b)

| | a4 | a3 |
|---|---|---|
| R1 | 3840 | 130 |
| R2 | 3830 | 135 |
| R3 | 3735 | 140 |
| R4 | 3725 | 129 |
| R5 | 3605 | 125 |
| R6 | 3570 | 139 |
| R7 | 3504 | 130 |
| G 2 | 3500 | 130 |
| R8 | 3449 | 140 |
| R9 | 3410 | 133 |
| R10 | 3205 | 139 |
| R11 | 3169 | 129 |
| R12 | 3140 | 125 |

Figure 11: Isomorphic graphs to G2 using a) EqualWidth and b) Clustering

## Conclusions and future work

As we can see, there are different unsupervised discretization methods that can be used to get the goal of our approach. In some cases we can choose different parameters, such as EqualWidth using Scott's rule, Freeman- Diaconis' rule, Birgé's rule or cross-validation. Similarly, in clustering methods we can use different similarity measures, which increase the number of possible combinations.

However, we can not try all possible combinations for each data set, because this would be computationally expensive. Therefore, we are currently performing different experiments in order to select some methods, able to cover as much as possible most of the data distributions, trying that the number of intervals will not be too large with respect to the data set.

On the other hand, we are exploring different techniques that allow us comparing the methods (Evaluation function) considering the number of necessary intervals, maximizing the number of data elements in each interval and minimizing the dispersion for each interval.

The SI-COBRA-C algorithm implements a support for continuous and discrete labels of vertices, which allow us work with a larger number of domains formed by such type of labels.

Another purpose is to work with real world databases to solve a specific problem, such as the detection of cancer cells or problems that focus in the detection of specific structures within a set of graphs.

## References


Agre, G., and Peev, S. 2002. On supervised and unsupervised discretization. *Cybernetics And Information Technologies* 2(2):43 – 57.

Biba, M.; Esposito, F.; Ferilli, S.; Mauro, N. D.; and Basile, T. M. A. 2007. Unsupervised discretization using kernel density estimation. *Proceedings of the 20th International Joint Conference on Artificial Intelligence* 696–701.

Birgé, L., and Rozenholc, Y. 2002. How many bins should be put in a regular histogram. *Technological Report, Laboratoire Probabilits et Modles Alatoires, Universit Pierre et Marie Curie, Paris, France*.

Borgelt, C., and Berthold, M. R. 2002. Mining molecular fragments: Finding relevant substructures of molecules. *In Proceedings of the 2002 International Conference on Data Mining (ICDM02* 51 – 58.

Boulle, M. 2005. Optimal bin number for equal frequency discretizations in supervized learning. *Intelligent Data Analysis* 9(2):175 – 188.

Cook, D. J., and Holder, L. B. 1994. Substructure discovery using minimum description length and background knowledge. *Journal of Artificial Intelligence Research* 1:231–255.

Cook, D. J.; Holder, L. B.; and Djoko, S. 1995. Knowledge discovery from structural data. *Journal of Intelligence and Information Sciences* 5(3):229–245.

Dougherty, J.; Kohavi, R.; and Sahami, M. 1995. Supervised and unsupervised discretization of continuous features. *In Proceedings of the Twelfth International Conference on Machine Learning* 194 – 202.

Freedman, D., and Diaconis, P. 1981. On the histogram as a density estimator: L2 theory. *Zeitschrift for Wahrscheinlichkeitstheorie und verwandte Gebiete* 57(4):453 – 476.

Ghazizadeh, S., and Chawathe, S. S. 2002. Seus: Structure extraction using summaries. *In Proc. of the 5th International Conference on Discovery Science*.

Han, J., and Kamber, M. 2006. *Data Mining, Concepts and Techniques*. Morgan Kaufmann Publishers, 2nd edition.

Inokuchi, A.; Washio, T.; and Motoda, H. 2000. An apriori-based algorithm for mining frequent substructures from graph data. *In Proceedings of the 2000 European Symposium on the Principle of Data Mining and Knowledge Discovery (PKDD00)* 13 – 23.

Kuramochi, M., and Karypis, G. 2002. An efficient algorithm for discovering frequent subgraphs. *Technical Report, Department of Computing Science, University of Minnesota*.

Nijssen, S., and Kok, J. N. 2004. A quickstart in frequent structure mining can make a difference. *In Proceedings of the 2004 ACM SIGKDD International Conference on Kowledge Discovery and Data Mining (KDD04)* 647 – 652.

Olmos, I.; Gonzalez, J. A.; and Osorio, M. 2005. Subgraph isomorphism detection using a code based representation. *Proceedings of the 18th International FLAIRS Conference* 474–479.

Schmidberger, G., and Frank, E. 2005. Unsupervised discretization using tree-based density estimation. *In Proccedings 9th European Conference on Principles and Practice of Knowledge Discovery in Databases* 240–251.

Scoot, D. W. 1979. On optimal and data-based histograms. *Biometrika* 66(3):605 – 610.

Sturges, H. A. 1926. The choice of a class interval. *J. American Statistical Association* 65 – 66.

Wörlein, M.; Dreweke, E.; Meinl, T.; and Fischer, I. 2006. Edgar: the embedding-based graph miner. *In Proceedings of the International Workshop on Mining and Learning with Graphs* 5:221 – 228.

Yan, X., and Han., J. 2002. gspan: Graph based substructure pattern mining. *Proceedings of the IEEE, International conference on Data Mining* 51–58.

Yang, Y., and Webb, G. I. 2002. A comparative study of discretization methods for naive-bayes classifiers. 159–173.