



**I
N
A
O
E**

Numeric Ranges Handling for Graph Based Knowledge Discovery

Oscar E. Romero A., Jesús A. González B., Lawrence B. Holder

Reporte Técnico No. CCC-08-003
27 de Febrero de 2008

© 2008
Coordinación de Ciencias Computacionales
INAOE

Luis Enrique Erro 1
Sta. Ma. Tonantzintla,
72840, Puebla, México.



Numeric Ranges Handling for Graph Based Knowledge Discovery

Oscar E. Romero A.¹ Jesús A. González B.² Lawrence B. Holder³

Computer Science Department
National Institute of Astrophysics, Optics and Electronics
Luis Enrique Erro # 1, Santa María Tonantzintla, Puebla, 72840, México
^{1,2}oromero,jagonzalez@inaoep.mx
School of Electrical Engineering and Computer Science
Washington State University
PO Box 642752, Pullman, WA 99164-2752
³holder@wsu.edu

Abstract

Nowadays, graphs based knowledge discovery algorithms do not consider numeric attributes (they are discarded in the preprocessing step or they are treated as alphanumeric attributes with an exact matching criterion), with the limitation to work with domains that do not have this type of attributes or finding patterns without numeric attributes. If we solve this problem for graph based algorithms; the knowledge discovery in databases process will allow finding classification and descriptive patterns using numeric ranges. In the classification task, if the algorithm does not deal with numeric attributes and one of them is the only one that helps to distinguish among classes, it will not be possible to find the best model with the available attributes. Opposite to this, if the algorithm possessed this capacity, it would be able to find the best model (if such a model is considered in its hypothesis space). For the description task, the power of expression of the model's patterns would be increased, being able to get more understandable, complete, and descriptive models.

1 Introduction

Knowledge discovery in databases is the process used to extract previously unknown, non trivial, and useful patterns from databases [6]. This information is extracted with advanced artificial intelligence and statistical techniques. The success of knowledge discovery depends on several factors such as data preprocessing and knowledge representation.

Data representation is one of the most important issues in artificial intelligence; it allows us to store descriptions of abstractions about the world so that they can be used by computer processes. These representations can then be used to describe what is known about a domain (a specific abstraction related to a domain such as a geographical database, or chemical compounds). Usually, entities (such as a tree or a house) are understood as objects, and objects are described through attributes (such as color, size, and so on), and there are also relationships among these objects (i.e. the tree is in front of the house).

In this way, we can describe part of the world in space and time using an adequate representation. A knowledge representation will also provide a mechanism that allows us inferring with the knowledge it

stores. There are different knowledge representation schemes (i.e. logic, graphs, etc.) and there are different ways to represent an object or a set of related objects using one representation (i.e. there are different ways to represent an object using graphs). In this thesis we work with a graph-based data representation that is described in the methodology section.

Different types of data (data belonging to particular domains) can be classified according to their type. There are flat domains that are used to describe objects with a set of attributes. There are also sequential domains that are described by a sequence of elements in a specific order; and finally, there are structured or complex domains where a set of objects is described as flat structures but we also consider the existent relations among some of these objects. These relations can be of different nature such as spatial, related to time, or physical properties of the objects, among others.

Given the previous data classification, a flat (attribute-value) domain consists of a set of records related to abstract representations of events and each event is composed of a set of specific values describing it (attributes). Some flat domains can be observed in commercial and scientific systems, where the events are stored in a database. Events can describe the employees of a company, the earthquakes in a specific area, or animals belonging to a habitat.

A sequential domain is represented by a group of examples which are described by a sequence of values. Examples of sequential structures that we can find in nature are related to molecular biology and biochemistry, such as DNA in the form of a sequence of nucleotides, amino acids, or proteins. Other examples of sequential (although artificial) domains are human language and written music [31].

Considering the above mentioned, a flat domain might become structured if each object (or some objects) are somehow related. Now we need to represent these relations. A straight forward way to represent these relations is using graphs where objects can be represented by vertices and the relations between objects are represented with edges. We can use labeled graphs to give a name to objects and their relations, for example, we can use two vertices to represent a tree and a house respectively and an edge labeled with the name of an existing relation between these two objects such as “in front of”. This is how we can represent flat, sequential, and structural domains with graphs.

Recently, artificial intelligence techniques have become a tool to solve real world problems and the need to represent real world domains has increased. Many interesting real world domains have an inherently structured character for which we need to find data representations to effectively apply AI algorithms to them.

We have now classified data domains as flat, sequential, and structural according to their properties. We can also divide data domains according to the characteristics of the attributes describing their objects (and relations for the case of structural domains). Using this criterion there are domains containing attributes with nominal values, those that contain characteristics with continuous (numerical) values, and those that contain both types of attributes, continuous and nominal.

In this sense, many algorithms have been developed to manipulate domains with these data types. As an example, graph based systems have been used to work with flat, sequential, and structured data [4]. Graph-based algorithms have been successfully used with acceptable results with nominal valued domains. Domains with continuous values are not appropriately manipulated by graph based knowledge discovery systems although they can be appropriately represented. There has not been developed a graph based knowledge discovery algorithm to deal with continuous valued attributes.

A solution proposed in the literature to approach this problem is the use of discretization techniques as a pre-processing or post-processing step but not at the knowledge discovery phase, however; we think that these techniques do not use all the available knowledge that can be taken advantage of during the processing phase. Thus, adding the capacity of dealing with continuous values would be a great contribution to the area

of graph-based knowledge discovery that would allow finding more precise patterns containing continuous valued attributes in the form of numerical ranges.

In order to achieve this goal, we will carry out a study of different approaches to work with numerical values to identify ideas that have worked satisfactorily in other applications and take advantage and use them in our graph-based algorithms. We will study the use of different data representations, clustering algorithms, discretization algorithms, algorithms for numerical numerical ranges manipulation, and attribute selection algorithms. We will analyze all these ideas to be able to extract useful concepts for the creation of a graph based algorithm able to handle numerical ranges.

2 Related Work

2.1 Discretization

Continuous values have important roles in data mining and knowledge discovery. These are created as intervals of numbers which are more concise to represent and specify, and easier to use and comprehend (as they are closer to a higher knowledge-level representation) than continuous values [32]. Data usually appears in nature in a mixed format, as nominal and continuous attributes, as we describe below.

- **Continuous** values are in the domain of real or integer numbers such as temperatures that can be categorized when we find a cut value to assign their values to the corresponding category (i.e. high or low temperature). This process is also known as discretization.
- **Nominal.** Nominal attributes define categories according to their meaning (i.e. eyes-color: black, blue, green, and brown).

Discrete characteristics coming from numerical data as in Temperature with the values high and low are closer to a higher level data representation, which are reduced or simplified through the discretization process. Dealing with discrete characteristics is easier, either for the classification or the description task. Discretization makes the learning process faster and precise.

2.1.1 The Discretization Process

Discretization can be univariable or multivariable. Univariable discretization quantifies one continuous feature at a time while multivariable discretization simultaneously considers multiple features [3]. The discretization process consists of four steps:

- ① Sorting the continuous values of the feature to be discretized,
- ② Evaluating a cut-point for splitting adjacent intervals or for merging them,
- ③ According to some criterion, split or merge intervals of continuous values,
- ④ Finally, stop at some point. In the following we discuss these four steps in more detail.

Sorting. The continuous values of a feature are sorted in either descending or ascending order. It is important to speed up the discretization process by selecting suitable sorting algorithms.

Choosing a cut-point. After sorting, the next step in the discretization process is to find the best “cut-point” to split a range of continuous values or the best pair of adjacent intervals to merge.

Splitting/Merging. In a top-down approach, intervals are split while for a bottom-up approach intervals are merged.

Stopping criteria. A stopping criterion specifies when to stop the discretization process. It is usually governed by a trade-off between lower arity with better understandability but less accuracy and a higher arity with a poorer understandability but higher accuracy.

There are numerous discretization methods available in the literature. These methods can be categorized according to the way they work. These can be dynamic vs. static, local vs. global, splitting vs. merging, direct vs. incremental, and supervised vs. unsupervised. We can construct different combinations of these levels to group the methods.

Each discretization method found in the literature discretizes a feature by either splitting the interval of continuous values or by merging the adjacent intervals. Both, the splitting and merging categories can be further grouped as supervised or unsupervised depending on whether class information is used or not.

Choosing a suitable discretization method is generally a complex task and depends on the need and other user considerations as well as the type of data that we want to discretize.

The discretization method can be used for the generation of numerical ranges since it is a very similar process, it generates data groups that can be seen as data ranges, the disadvantage is that the discretization process is used in the pre-processing phase; and the numerical ranges are not modified during the Data Mining process as we propose in this work.

2.2 Clustering

Clustering consists in a division of the data in groups based on the similarity of the objects. It is often used to describe a new groups of data, which can be assigned a new label. A process that finds few groups loses details, but it simplifies the representation. We prefer few groups with a better discrimination power.

Groups are not necessarily similar among them (actually we want groups to differ as most as possible) but sometimes there might be overlap among some of them.

Clustering is the unsupervised classification of patterns (observations, data items, or feature vectors) into groups (clusters), clustering is a difficult combinatorial problem .The steps of the clustering process can be described as [25]:

- ① Data representation including the features extraction and selection steps.
- ② Select a distance measure appropriate to the data domain.
- ③ Select the clustering method.
- ④ Select the type of data abstraction (i.e. method to label the groups/classes and choose the evaluation measure for the created groups)
- ⑤ Select the type of evaluation according to the task: classification or description.

Clustering algorithms are classified according to these criteria:

- ① The form in which the cluster is generated
- ② The structure of the data
- ③ The distance function used to create the groups

The different types of clustering algorithms can be divided depending of how the data is processed [25].

- ① In the first division we have hierarchical versus partitioning methods. A hierarchical method, generates small groups that have a pattern and sequentially construct a binary tree. The binary tree establishes a relation and the length between edges is the distance between the nodes. Partitioning methods, initially group all the information in only one cluster and successively separates them until each of them is isolated as an entity. They construct several partitions and evaluate them following some criterion.
- ② Agglomerative vs. Divisive: An agglomerative approach begins with each object in a distinct cluster, and successively merges clusters together until a stopping criterion is satisfied. A divisive method begins with all objects in a single cluster and splits them until a stopping criterion is met.
- ③ Mono-variable vs. Poly-Variable: The poly-variable algorithms uses all features for the computation of distances between objects, and decisions are based on those distances. A mono-variable algorithm only considers one feature to divide the given collection of objects.
- ④ Hard vs. Fuzzy: A hard clustering algorithm allocates each object to a single cluster. A fuzzy clustering method assigns an object degrees of membership to several clusters.
- ⑤ Incremental vs. Non-incremental: This issue arises when the set of objects to be clustered is large, and constraints on execution time or memory space affect the architecture of the algorithm.

In clustering, the characteristics of the object are considered to generate a set of groups, for which the distances between objects are calculated (considering all object attributes). In our work we want to use the clustering distance measures to generate our numerical ranges where each range would correspond to a cluster and the cluster centroid is the central point of the range.

2.3 Graph-Based Algorithms and Knowledge Discovery

Graph-based data mining systems can be grouped according to the way in which the graphs are built (these graphs are built using a finite search space usually defined by the input graph, more specifically the input vertices and edges). The construction of these graphs is generally known as expansion, which can be done using a candidate generation or a non-candidate generation approach.

During the non-candidate generation expansion process (gSpan [9]), the key problem consists in deciding which vertex should be expanded. This decision is generally based on the number of occurrences (support or number of instances) of the new substructure using the input graph. The idea is to successively expand the vertex that has more instances until a substructure with the desired size (number of vertices and edges) is found.

In the case of candidate generation expansion (Apriori [18], AGM [20], FSG [26], Subdue [15], and SEuS [11]), a set of initial substructures to expand is first found (usually a set of substructures with only one vertex, each vertex representing a unique label of the input graph) and then these are expanded (each of these candidate substructures is expanded by adding an edge or and edge and a vertex). After that, the candidate substructures are evaluated (using some evaluation metric as in the case of Subdue that uses the MDL principle) in order to choose the best of them (for this, it is necessary to look for the instances of each candidate substructure in the input graph).

Both of these expansion processes (with or without candidate generation) use the input graph to search for instances of the substructures to be evaluated. This step is generally based on graph matching functions, which can be either exact or inexact.

An exact graph matching function (AGM [20], FSG [26], SEuS [11], Subdue [15], SI-COBRA [28]) determines if two graphs are equal (considering vertices, edges, and their structure). The process that verifies the equality between two graphs is known as graph isomorphism [5], the process that verifies if a graph is contained in other graph is known as subgraph isomorphism.

A graph isomorphism test (Nauty [27]) verifies for two given graphs **A** and **B** if there exists a one to one mapping among the vertices and edges of graph **A** to the vertices and edges of graph **B** (a bijective function) keeping the same structure. For the case of labeled graphs, the previous definition is extended to also verify that the labels (in vertices and edges) are also equal.

The subgraph isomorphism test (Ullmann's algorithm [34]) verifies for two graphs **A** and **B**, (where **A** is smaller than **B**) if all the graph vertices and edges of graph **A** can be mapped to graph **B** (this means that graph **A** is included in graph **B**), including identical labels either for vertices as for edges and keeping the structure of graph **A**.

The inexact graph match [9] (Subdue [15], SI-COBRA [28]) is necessary when there exist small differences among the graphs being evaluated but the graphs can still be considered to match. These differences can be found in the values of vertices or edges or in the structure of the graphs and can be due to domain characteristics (the nature of the data, variations of the data) or to noise that could not be detected in a cleaning phase. Some of the graph-based algorithms described in this section are related to the apriori algorithm described below.

2.3.1 The Apriori Algorithm

Apriori [18] is a method used to generate association rules, it generates candidates at each iteration from the candidates created for the previous iteration. The idea consists of combining two candidates (graphs in our case) of size n that share all their items (vertices in our case) but the last one to create a new candidate of size $n+1$ with the idea that if the two candidates were frequent there is a high probability that the new candidate will also be frequent. There is also the possibility of discarding (pruning) candidates that we know will not be frequent (this frequency is measured with the number of instances of the candidate in the input graph). The original apriori algorithm has been modified in different ways to run faster [18]. The candidates generation used in apriori has also been used to create graph-based candidate generation algorithms such as AGM [18] and FSG [26].

2.3.2 The AGM Algorithm

The Apriori - based Graph Mining (AGM) algorithm ([18], [19]), is a proposal based on the Apriori strategy, which uses adjacency matrices to represent the graphs. This algorithm tries to decrease the dimension of the expansion set (the set of candidates) introducing a linear order for the vertices' labels of the graph. Due to this representation, AGM only works with simple graphs (simple graphs do not allow neither loops nor more than one edge between a pair of vertices) and only allows labels in vertices (not for edges). The way in which AGM creates candidates is by combining two adjacency matrices that only differ in one column or a row. Finally, the graphs (adjacency matrices) are evaluated according to their support in the input graph.

2.3.3 The FSG Algorithm

The Frequent Sub-Graph discovery (FSG) Algorithm [26] is also based on the candidates generation approach. This method finds all the connected sub-graphs from a set of simple and not directed graphs (the input graph with an adjacency matrix representation). It uses a stepwise expansion strategy, adding an edge at a time. This algorithm calculates the frequency of each label and uses it to prune those candidate graphs that will not have the minimum support without the need to verify it in the input graph.

As we mentioned AGM and FSG generate candidates by combining structures that were already identified as frequent (in a previous iteration). They do this with the idea that combining these substructures will produce a new one with a large probability to have the minimum support (number of occurrences). The problem here is that there might be multiple ways to combine candidates to represent the same graph, which takes extra processing time that can be reduced as shown in other methods such as gSpan.

2.3.4 SEuS: Identification of substructures using summaries

Structure Extraction using Summaries (SEuS) [11] is a graph-based algorithm which strategy consists of avoiding the access to the database (the input graph) in order to determine the support of common structures. It generates a set of summaries or abstractions of the graph, which are used for the generation and support estimation of common structures. SEuS is implemented in three phases. In the first phase it pre-processes the database (input graph) to produce the summaries. In the second phase it generates a set of candidates, which are obtained by expanding the common structures generated in the previous iteration (adding an edge at a time). These expansions are done according to the summaries created at the beginning of the process, which avoids accessing the original database. Finally, in the third phase it obtains the exact support of the structures found in phase two. In this way, SEuS delays the access to the input graph until it finds a set of promising structures. This algorithm uses an economic expansion process carried out from the frequencies of each vertex and edge of the input graph. SEuS is able to find all the substructures of the input graph.

2.3.5 The gSpan Algorithm

gSpan [37] is another graph-based system that finds all the existing substructures from an input graph, restricted to simple and non-directed graphs. This system uses a representation based on lists of codes, which are expanded step by step according to the existing patterns of the input graph (this means that it does not generate candidates). Each code of a list of codes represents the relation vertex-edge-vertex. The lists are called DFS (Depth First Search) codes, which are expanded by adding an input vertex at the time. The expansion is directed by a set of rules, which restrict the type of edge that can be added using a depth first search. The lists of codes that represent our graphs will be used to compare them and obtain common substructures. However, even with the construction rules, it is possible to obtain several DFS codes that represent the same graph. This is why gSpan adds a lexicographic order to the set of DFS codes and chooses the lowest to represent the graph. These are the codes that are finally used to identify common substructures. This strategy has several advantages that make the comparison among graphs faster (besides, it does not generate candidates and implements pruning routines) but its main disadvantage is that it repeatedly accesses the input graph.

2.3.6 Multivariable Discretization by Recursive Supervised Bipartition of Graphs

In our search for related work we found an algorithm for multi-variable discretization that works with a graph-based data representation. In this research by Sylvain Ferrandiz and Marc Boullé [8] they presented a supervised, local, static, multi-variable and weakly biased graph partition algorithm. They propose an evaluation criterion of bi-partitions, which is based on the Minimum Description Length (MDL) principle and apply it recursively. This is an example of how it is possible to apply multi-variable discretization using the properties of the graph that they build.

For the uni-variable case, they first consider the frequencies of each label of each class in a different interval, and with this information they sort the labels of each class. To determine the information gain, they use Shannon's entropy, to sort the classes labels.

For the multi-variable case, they sort the data, and for each variable they create groups that represent the intervals by means of the uni-variable process and later they calculate the entropy of each interval. For this case it is necessary to consider a distance measure between two sequential labels to determine if they are adjacent, they use the Gabriel distance measure (in the method to obtain the Gabriel adjacency measure, a circle which perimeter touches both points A and B is drawn), which is the minimal number of edges necessary to connect two sequential labels. The way in which they handle the properties of the graph, based on the MDL principle is a good idea, though the authors do not directly work with the graph, they use the graph properties to define a discretization metric.

2.3.7 Subdue

Subdue is a graph-based knowledge discovery system developed at the University of Texas at Arlington by the machine learning group [14]. Subdue has been successfully used in different structural domains such as DNA [7]. The input graph to Subdue is a graph that represents objects and their attributes with labeled vertices and edges, vertices labels give names to these objects or their attributes. Relations between objects and relations among objects with their attributes are represented by labeled edges, where the label determines the name of the relation. This allows us working with any domain that can be represented with graphs.

Subdue's evaluation model decides which patterns are going to be chosen as important. For this, Subdue implements two evaluation criteria.

The first evaluation method is called "Minimum Encoding", which is a variant of the minimum description length principle (MDL) introduced by Rissanen [30]. This method chooses as best substructures those that minimize the description length metric that corresponds to the length in number of bits of the graph representation. The number of bits is calculated based on the size of the adjacency matrix representation of the graph. According to this, the best substructure is the one that minimizes $I(S) + I(G | S)$, where $I(S)$ is the description length of the input graph and $I(G | S)$ is the number of bits required to describe graph G after being compressed by substructure S [13].

The second method chooses the best substructures according to how well they compress the graph in terms of their size in number of vertices and edges. Subdue uses a computationally constrained beam search. The search begins with the substructures matching a single vertex in the graph. Each iteration, the algorithm selects the best substructure and incrementally expands the instances of the substructure in all the possible ways. The evaluation of each substructure is determined by how well the substructure compresses the input graph according to the heuristic being used (MDL or Graph Size). The best substructure found by Subdue can be used to compress the input graph, which can then be input to another iteration of Subdue.

Subdue is also able to perform the concept learning task and this version of the algorithm is known as SubdueCL [12]. SubdueCL accepts a set of positive and negative examples as its training set and uses a set covering approach to find substructures that describe a sub-set of positive examples but not negative examples. Then, the algorithm learns a set of rules in DNF [16]. Each iteration, SubdueCL chooses the substructure that covers a large number of positive examples without covering negative examples (minimizes the number of negative examples covered). The positive examples covered by a substructure are then removed from the training set and the process is repeated with the rest of the examples until all the positive examples have been covered.

The MDL Heuristic

Subdue’s heuristic for evaluating substructures is based on the Minimum Description Length (MDL) principle, which states that the best theory to describe a data set is that theory which minimizes the description length of the entire data set. The concept’s description length is the number of bits in the bit string.

This compression is calculated as [4]:

$$Compression = \frac{DL(S) + DL(G|S)}{DL(G)} \quad (1)$$

where $DL(G)$ is the description length of the input graph, $DL(S)$ is the description length of the substructure, and $DL(G | S)$ is the description length of the input graph compressed by the substructure. Subdue has a way to deal with an inexact matching in vertices and edges.

Inexact Numeric Label Match

Labels may represent numeric values in this case, the graph match algorithm is modified to allow some variation in these values for the corresponding vertices and edges labels.

The three allowed options are [2]:

- ① **Exact Match:** Label X matches another label Y if $X = Y$.
- ② **Tolerance Match:** Label X matches Y if the absolute value of their difference is less than t , a threshold parameter.
- ③ **Difference Match:** $MatchCost(X, Y)$ is defined as the probability that Y is drawn from a probability distribution with mean of X and standard deviation defined in the input file.

When Subdue is run on input graphs with numeric labels, these parameters are used to decide when two numeric labels can be considered to match. The previous options are a very good approximation to the non identical comparisons of the labels, but they have not given good results because they need domain knowledge (from a human expert) to be able to choose the ideal parameters so that subdue can find useful substructures.

Post-processing of Numeric Patterns

This is a post-processing method applied by Subdue when it prints the discovered substructures. For each discovered substructure it finds all its similar instances with small variations in their labels (either in edges or vertices). These local numerical ranges are given by the variations of the values of the discovered substructures.

Subdue creates a substructure definition for each discovered pattern, these substructures definitions need to provide the range of values covered by instances of the substructure in the graph [22].

This method is an excellent solution to join the substructures found (substructures that slightly differ in their numerical labels can be represented by a numerical range for each of those numerical labels). This variation of the labels is local because they fix part of the substructure and find variations of the leaf nodes that correspond to numerical labels (for each different attribute of the substructure); this process works better when the labels are identical. This differs from our proposal because we will use numerical ranges to discover new substructures (during the processing phase).

Clustering

Subdue was extended to perform the clustering task, this version of Subdue is called SubdueGL, which constructs a hierarchical lattice of clusters, the substructure discovered after each iteration comprises a cluster [23]. This cluster is inserted into the classification lattice and used to compress the input graph. The compressed graph is passed again to Subdue to find another substructure. This iteration allows Subdue to find new substructures defined in terms of previously discovered substructures.

In order to find the cluster in the input graph, it uses the variable concept, which is described in the following section.

Variables

A variable is a label that can be a vertex or an edge. The first step towards discovering variables is discovering commonly-occurring substructures, since variability in the data can be detected by surrounding data. If commonly occurring structures are connected to vertices with varying labels, these vertices can be turned into variables [24].

SubdueGL discovers variables inside the ExtendSubstructure search operator. As mentioned before, SubdueGL extends each instance of a substructure in all possible ways and groups the new instances that are alike. After this step, it also groups new instances in a different way. Those that were extended from the same vertices by the same edge in all instances, regardless of what vertices they point to, are grouped together [21]. Let $(\mathbf{v1}, \mathbf{e}, \mathbf{v2})$ represent edge \mathbf{e} from vertex $\mathbf{v1}$ to vertex $\mathbf{v2}$. Vertex $\mathbf{v1}$ is part of the original substructure which was extended by \mathbf{e} and $\mathbf{v2}$. For variable creation, instances are grouped using $(\mathbf{v1}, \mathbf{e}, \mathbf{V})$, where \mathbf{V} is a variable (non-terminal) vertex whose values include the labels of all matching $\mathbf{v2}$.

The previous concepts (clustering, and variables) used by SubdueGL to find substructures, differ from our proposal because the variables are considered local to the substructure discovered at each iteration. We will work with dynamic numerical ranges at each iteration that consider all the values of each label in that iteration (the input graph of each iteration is different because of the graph compression or set covering approach being used) to generate the numerical ranges sets, additionally in our work we will implement new metrics that consider comparisons between numerical ranges and between numerical ranges vs numerical values.

2.3.8 Analysis

Currently, Subdue is able to perform many tasks but it can be enhanced to work with real world domains containing mixed data (nominal and continuous), to find interesting and unknown patterns (substructures that include dynamic numerical ranges), for the tasks of classification and concepts description.

For this work we will use the minimum description length MDL evaluation heuristic, because it has given significant results, and it does not need from domain knowledge to work properly. This evaluation heuristic considers that the best substructure (pattern) is the one that better compresses the input graph. Up to now we

have not considered a modification of this evaluation heuristic, but we might need to do it. In addition we are considering that the numerical ranges generated should be used either for the concepts description (MDL) as for the classification task (Set Covering). Therefore, we might also need to modify the set covering evaluation measure.

We also need to consider a modification to the subgraph isomorphism algorithm used inside the substructures search process, because the substructures (patterns) will have numerical ranges. This modification will be used for the comparison of these new type of numerical labels.

The inexact numerical label match of subdue will not be used in this work, it will be replaced with the numerical ranges handling that we propose. We decided to replace it because as we described before, it is difficult to use due to the need of domain knowledge to choose subdue’s input parameters. It is very difficult to find the right parameters that make subdue find interesting substructures.

We cannot use the current Subdue’s comparison functions in our implementation for the numerical ranges handling, because at each new iteration we will have new labels that correspond to our numerical ranges, which will be compared with other numerical ranges, or with a numerical label. Then, it is necessary to implement a new comparison function, which must consider the points exposed in the methodology (comparisons between numerical ranges, between numerical labels and between numerical labels and numerical ranges).

Subdue has an implementation to perform the conceptual clustering task (SubdueGL). This implementation uses a post-processing of the substructures (when it prints the found substructures) which is used to join the substructures that belong to the same cluster (based on local values of the variable), using the concept of variables discovery, in which a range of possible values can arise for each variable. These substructures are then given to Subdue as predefined substructures to find their instances. This implementation allows finding substructures with small variations in their labels, grouping them in a cluster, which allows the cluster to grow in a post-processing phase (local to each variable) and therefore, it involves a **new search** of the substructures. This consumes time and resources, because the substructures that will be expanded are substructures that were already found and Subdue will not discover new knowledge.

As we can see, this work differs with others because we consider numerical ranges during the search of the substructures (in the processing phase), not as a data post-processing, in such a way that the discovered substructures may contain numerical ranges labels. The numerical ranges used at each iteration are dynamic (regenerated in each iteration), which must generate better substructures (more useful for the classification task), which involves an extension to the Subdue algorithm.

Although decision trees are not a graph-based algorithm we will use them to compare the results of our graph-based algorithm. Its is important to note that the graph based algorithms described before are not able

Method	Type of Graph	Type of Expansion	Complete	Available Code	Numerical Attribute
AGM	Simple (no)directed	Generation of candidates	No	No	No
FSG	Simple label (no)directed	Generation of candidates	Yes	No	No
gSpan	Simple no directed	No generation of candidates	Yes	No	No
SEuS	Simple label (no)directed	Generation of candidates	Yes	No	No
SUBDUE	General	Generation of candidates	No	Yes	No

Table 1. Graph-Based Algorithms.

to deal with numerical attributes without using a discretization during a preprocessing phase.

Table 1 shows a summary of the characteristics of the graph-based algorithms described in this section according to the type of graphs that they use, the type of expansion that they follow, whether they are complete or not, if there is code available to be able to compare with them and finally we show if they are

able to deal with numerical attributes. As we can see from this table, none of these graph-based algorithms currently deal with numerical attributes in the way we described.

2.4 Decision Trees

Decision trees are one of the most popular methods used for inductive inference. They are robust for noisy data and capable of learning disjunctive expressions. A decision tree is a k -ary tree where each of the internal nodes specifies a test on some attributes from the input feature set used to represent the data. Each branch descending from a vertex corresponds to one of the possible values of the feature specified at that vertex. The basic algorithm for decision tree induction is a greedy algorithm that constructs decision trees in a top-down recursive divide-and-conquer way. C4.5 [29] builds a decision tree using the standard TDIDT (top-down induction of decision trees) approach, recursively partitioning the data into smaller subsets, based on the value of an attribute. At each step in the construction of the decision tree, C4.5 selects the attribute that maximizes the information gain ratio. The induced decision tree is pruned using pessimistic error estimation.

The decision tree algorithm usually uses an entropy-based measure known as “information gain” (although other measures are also possible) as a heuristic for selecting the attribute that will best split the training data into separate classes. C4.5 algorithm computes the information gain of each attribute, and in each round, the one with the highest information gain will be chosen as the test attribute for the given set of training data. A well-chosen split point should help to split the data to the best possible extent.

We use this algorithm to compare our graph-based classification results when we use numerical ranges (static and dynamic).

3 Proposal

3.1 The problem

Graph based algorithms have been used for years to describe (in a natural way) flat, sequential, and structural domains with acceptable results [13]. Some of these domains contain important numeric attributes (attributes with continuous values). Although these domains can be represented with graphs, the algorithms that processes them do not treat these attributes in a special way and due to this, graph-based classification algorithms do not obtain results as effective as other algorithms that are able to deal with numeric attributes (such as the C4.5 [29] classification algorithm that although it does not work with structured domains, it does manage numeric attributes for flat domains). If we are able to provide graph-based algorithms with the capability to handle numerical ranges, we will be able to work with structured domains containing numeric attributes and in this way it could be possible to improve the classification accuracy and the descriptive power of the patterns found with graph-based algorithms for structured domains.

3.2 Motivation

In some research areas such as data mining and machine learning, the domain data representation is a fundamental aspect that determines in great measure the quality of the results of the discovery process. Then, when a domain can be represented in multiple ways, we need to find the best one through experimentation. Depending on the domain, the Data Mining process analyzes a data collection (such as files, log files, relational databases, etc.) to discover patterns, relationships, rules, associations, or useful exceptions to be used for decision making and for the prediction of events and/or concepts description.

Selecting the Data Mining algorithm to use for a specific domain is another important task, there are several factors in consideration such as the knowledge representation we need to use because of the nature of our data (as we mentioned, it can be flat, sequential, or structured), the Data Mining task that the algorithm approaches (such as classification, association rules, or summarization), and the data types (such as nominal, binary, or continuous valued) of the involved attributes so that the algorithm can thoroughly analyze the data. Then, we need to carefully study the Data Mining algorithms and data characteristics in order to take a good decision when choosing an algorithm to work with. If we select several algorithms, we choose the best one through experimentation or we may end using several of them. On the other hand, we also need to choose one or several representations and experiment over them to select the best one.

This mainly happens when working with structured domains. It is here where we want to center our research, providing graph-based algorithms with the capacity to deal with numeric attributes.

Numeric values, as any other attribute type, play an important role in the data mining and knowledge discovery process. These are usually generalized as numerical ranges when finding patterns and these can be represented in different ways to be used by knowledge discovery algorithms.

In our work we use graphs as our data representation. This decision was taken because of the easiness and flexibility that graphs provide as a way to describe structured domains. These structured domains, as expected, contain attributes with numeric values, but graph-based algorithms do not treat these numeric characteristics in a special way because they are considered as alphanumeric labels [21] (they do not identify that number 2.1 is similar to 2.2 and takes them as totally different values). This is the reason why these algorithms do not obtain so rich results as other algorithms that manage numeric attributes in a special way.

3.3 Justification

Up to now, graph based knowledge discovery algorithms do not consider numeric attributes (they are discarded in the preprocessing step or they are treated as alphanumeric with an exact match criterion), with the limitation to work with domains that do not have this type of attributes or finding patterns without numeric attributes. If we solve this problem for graph based algorithms; the knowledge discovery in databases process will allow finding classification and descriptive patterns using numerical ranges.

In the classification task, if the algorithm does not deal with numeric attributes and one of them is the only one that helps to distinguish among classes, it will not be possible to find the best model with the available attributes. Contrary to this, if the algorithm possesses this capacity, it would find the best model (provided it is considered in its hypothesis space). For the description task, the power of expression of its patterns would be increased, being able to get more understandable, complete, and descriptive models.

3.4 Hypothesis

Does the manipulation of numeric data for graph-based data representations in graph based knowledge discovery improves the predictive / descriptive capacity of the discovered patterns in comparison with the results reported in the literature for other graph based algorithms?

Would adding graph-based algorithms with the capacity to deal with numeric attributes allow them finding numeric patterns that are more understandable, and descriptive for the user when working with structural domains with mixed data types?

3.5 General Objective

Provide a graph-based Data Mining algorithm with the capacity of dealing with numeric attributes.

3.5.1 Particular Objective

- ① Extend a sub-graph-isomorphism algorithm to be able to deal with mixed attributes (continuous valued attributes)
- ② Create an algorithm for the generation of numerical ranges to feed a graph-based data mining tool that deals with continuous attributes.
- ③ Decide if we will use a candidate generation or non-candidate generation graph-based data mining algorithm.
- ④ Extend the selected graph-based data mining algorithm to deal with numeric attributes using the extended sub-graph-isomorphism and the numerical ranges generator algorithms.
 - ① In case of working with a Candidate Generation Algorithm
 - ① Extend the graph-based data mining algorithm to be a single-variable (only considers one numeric variable)
 - ② Extend the single -variable algorithm to be multi-variable
 - ② In case of working with a non-Candidate Generation Algorithm
 - ① Extend the graph-based data mining algorithm to deal with multiple variables
- ⑤ Create or use an attributes selection algorithm to choose the continuous attributes to be used in the graph-based data mining process.
- ⑥ Extend the graph-based algorithm with the capability of multi-variables numerical ranges handling with attributes selection, during a pre-processing phase.
- ⑦ Extend the graph-based algorithm with the capability of multi-variables numerical ranges handling with attributes selection, during a processing phase.
- ⑧ Extend the graph-based algorithm with the capability of multi-variables numerical ranges handling with attributes selection, during a post-processing phase.

3.6 Scientific Contribution

The main contribution of this work consists of the creation of a graph-based representation for mixed data types (continuous and nominal) and also the creation of an algorithm for the manipulation of these graphs with numerical ranges for the data mining task (classification and description). In order to accomplish the previous contributions, we need to extend a sub-graph isomorphism algorithm to give it the capacity to handle numeric attributes by means of numerical ranges.

Achieving the previous contributions will make us able to create a graph-based algorithm able to handle numeric attributes for the classification and description Data Mining tasks. This will also allow working with domains represented with graphs containing numeric attributes in a more effective way.

4 Methodology

In order to add to the graph based knowledge discovery system Subdue the capacity to handle numerical ranges, we propose a graph-based data representation (for this example of a flat domain we use a star graph but we will also work with structural domains), as we show in figure 1 for the attributes “Temp” with a value of “4.5” and “Hum” with a value of “3.2”, in which the vertex labeled as “Exa” is in the center of the graph and the attributes describing the example identified by vertex “Exa” are represented with vertices linked to it. We need to extend this representation to allow the use of numerical ranges as we show in figure 2 where attribute “Temp” now may have a value between “3.8” and “9.5” representing the range [3.8, 9.5] and attribute “Hum” may take values between “3.0” and “4.7” for the range [3.0,4.7]. It is important to note that having a graph-based data representation that allows using numerical ranges also needs as complement an algorithm that manipulates it for the description and classification tasks that we consider in this work. It is necessary to implement the corresponding graph operations (such as the subgraph-isomorphism algorithm) needed for the discovery phase. For the knowledge discovery task, we need to be able to manipulate data

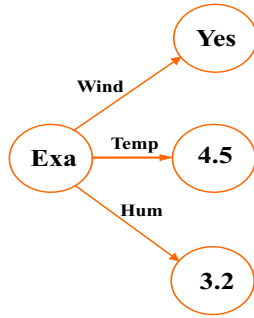


Figure 1. Vertices with Numerical Labels.

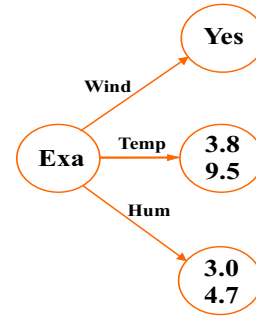


Figure 2. Vertices with Numerical Ranges.

with numerical ranges whether the domain is flat or structured. For this, we need to:

- Discover the occurrences of the substructures that contain numerical values.
- Identify those interesting substructures (for the user) that contain numerical values.

Since we look for regularities in the data, for the nominal attributes case, the problem consists of finding a model that describes the classes using the attributes values of the description. For numerical data the problem consists of finding a set of numerical ranges for each continuous variable such that these numerical ranges help to describe the class attribute.

For the classification task, if the algorithm does not handle numerical attributes and these are the attributes that better help to distinguish between the classes, then it is not going to be possible to find the best model given the available attributes, opposite to this, if the algorithm has this capacity, it is going to find the best model (if it belongs to its hypothesis space).

For the description task, the power of expression of the found patterns would be increased, obtaining understandable and descriptive models.

Dealing with numerical attributes is a complex task because of different reasons such as:

- The number of different values for continuous variables is infinite, which is almost always known by the user.

- A single domain may contain several continuous attributes.
- It would be good to consider the dependence or independence among the continuous attributes of a domain.
- It would also be good to consider the dependence or independence of the classes with the continuous attributes.

Given a continuous variable, there are different ways to generate numerical ranges from it (different ways to discretize it), the selection of the best set of numerical ranges (obtained with a specific discretization method) is based on its capability to accurately classify the dataset.

The evaluation algorithm must have the property of finding the lower possible number of cut points that generalizes the domain without coming to specialization. The problem now consists of finding the borders of the intervals and the number of intervals (or groups) for each numerical attribute.

Another problem that we have to consider is that there are databases with thousands of records and each of these records has many attributes (more than 20), then we may have storage space and execution time problems. Here the trivial process becomes complicated because of the quantity of data to process.

The most effective way of solving this is to consider subsets of information and attributes selection, in order to generate representative subsets of data for the domain. It is also necessary to think that most of these large databases have missing data, some of these records do not contain values for all their attributes, and many of these values can be erroneous, considered as noise.

Considering all these characteristics, our methodology for the numerical ranges handling is based on the knowledge discovery process and we focus in the graph-based data mining algorithm Subdue to implement the proposed solution. Figure 3 shows a modification of the knowledge discovery process so that it shows the methodology that we will follow in this thesis, and focus in describing details of our graph based approach (shows how we are going to solve the objectives of this thesis). In the data mining section we try with three schemes, which depend on where we are going to introduce the numerical ranges to Subdue. Figure 4 shows the scheme in which we use data pre-processing and later we apply the Subdue data mining algorithm and finally we evaluate the found substructures, this is similar to the discretization process.

Figure 5 shows the scheme in which we apply a postprocessing step, to find more instances of the reported substructures, this is similar to the work performed by SubdueGL (reported in the section of related work), but we are going to generate different sets of numerical ranges for each attribute (this is a proposal for the generation of numerical ranges), and later we evaluate which numerical ranges set find more instances and increase the accuracy of the final model. Figure 6 shows the scheme in which the generation of numerical ranges is performed during the data mining process, to generate dynamic numerical ranges that can find better patterns during the substructures expansion and search processes.

A very important part of our work is related to the Subdue system, which will be extended to deal with numerical ranges. Figure 7 shows how Subdue works, it receives an input graph (or set of graphs) and it looks in it for the best substructure in each iteration, the search process is described in more detail through figure 8.

When Subdue finds the best substructure it compresses the input graph by substituting the best substructure instances with one vertex, this process is also explained with more detail in the following paragraph, Subdue reports the best substructure and depending on the input parameters it continues with a new iteration until a termination condition is reached or the graph can not be further compressed. The processes described below are more detailed because the work of this thesis is directly related with them. Figure 8 describes the part of the algorithm that searches for the best substructure and the part that compresses the graph.

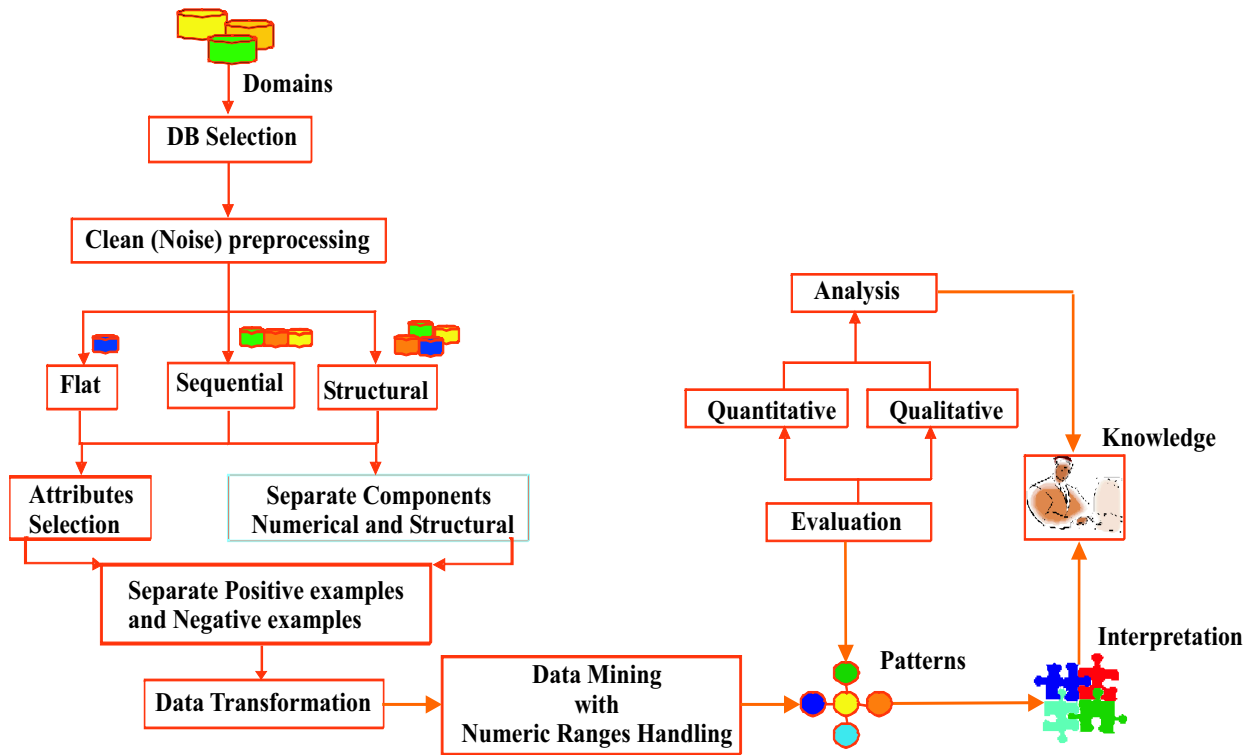


Figure 3. General Methodology of the Thesis.



Figure 4. Working with Numerical Ranges during a Preprocessing Phase.

Inside the graph compression process, the algorithm applies a subgraph isomorphism test which has three evaluation methods to determine when two sub-graphs are considered to match, the selection of the method to use is given by the user through parameters. We need to modify this process to perform an inexact graph match able to consider sub-graphs that could have in their labels the generated numerical ranges, this comparative process will include the following considerations:

- ① The basic comparisons (i.e. nominal labels, nominal vs numerical).
- ② Comparisons between numerical labels and a range, to determine if a numerical label is covered by the range.
- ③ Comparisons between two numerical ranges, due to the fact that there exist different ways of generating numerical ranges some methods could generate numerical ranges that are included in another, or numerical ranges with some intersection, or disjoint.



Figure 5. Working with Numerical Ranges in a Postprocessing Phase.



Figure 6. Working with Numerical Ranges during the Data Mining Processing Phase (Dynamical Ranges).

Figure 8 shows the process to search for the best substructure, in this section we need to add Subdue the capability to generate numerical ranges, then we will modify the “Expand substructures” module to consider the numerical ranges at the moment to perform the search. Once we have generated the numerical ranges using the six techniques that we describe at the end of this section we will determine which of them will be used. We will first try giving all of them to Subdue and let it decide through its evaluation process which of them will be selected in its substructure selection process or we could eliminate some of them according to a criterion such as entropy or overlapping among them.

In the following step Subdue generates a substructure from each unique label contained in the input graph including the new generated numerical ranges, in such a way that now, those unique labels, may be numerical ranges.

The following step consists of the expansion in the initial substructures. Traditionally, Subdue expands each substructure by adding an edge (for substructures containing at least two vertices) or an edge an a vertex. Now this expansion may connect a substructure with a vertex with a numerical range.

We now order the substructures by their number of instances. The substructure with more instances will be considered as the best. We then evaluate the substructures.

After we evaluated our substructures they are ordered (according to the used evaluation method). It is important to note that Subdue may use two main evaluation methods. The first uses the MDL principle to compress the graph (which works as an unsupervised version) and the other works with a set covering approach (supervised), then there are two ways in which the substructures can be ordered. The ordered substructures are stored in a list and we reject those that do not satisfy the input parameters (such as the beam size). Subdue uses a beam search algorithm. The default value for the beam is 4 but it can be modified in the input parameters.

When the termination condition is reached the best substructure is reported and used to compress the input graph to be used in the next iteration.

In figure 9 we show the process to evaluate a substructure with Subdue. The evaluation method is given to Subdue as an input parameter, it can be MDL, Size, Set Cover, and probably the modification to MDL that we might add.

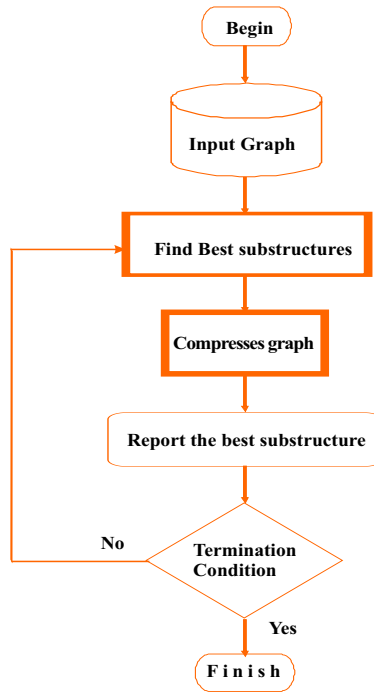


Figure 7. Subdue's General Process with Numerical Attributes.

Subdue reports a value that tells us how the best substructure performed with the specific evaluation method used and its number of instances. It stores the substructure in a list of best substructures (limited by the beam size and ordered by the evaluation value).

For our modified MDL evaluation, Subdue will find the substructure instances in the input graph and calculate its MDL value as if this substructure were used to compress the graph (the input graph is not really compressed until the best substructure is found, this is just a temporal processing used to evaluate substructures).

For the numerical ranges creation (as shown in figure 9) we will implement several techniques to generate numerical ranges. These are based on the nature of the data. We will later describe the numerical ranges generation block included in figure 8.

The idea is to implement several techniques for the generation of numerical ranges, these techniques must be used in the substructures expansion process and we also need a comparison heuristic (match between numerical ranges or a numerical range with a numerical label).

At this moment the numerical ranges are created statically (we generate them outside Subdue, as a pre-processing step) and we want to incorporate this process within Subdue, we will show in our preliminary results, evidence that tells us about the improvements that can be achieved through the dynamic creation of numerical ranges. Currently we are trying modify Subdue to add the new numerical ranges heuristic.

The difference between these two algorithms is that the first algorithm considers frequency histograms to generate centroids and the other algorithm sorts the values (a modification radix sort) and groups them according to their proximity. The distance between the values is calculated by any of six methods. We have six ways of calculating the distance between a set of elements, these distances are, a modification to the Tanimoto distance [17], a modification of the Euclidean distance [35], a modification to the Manhattan

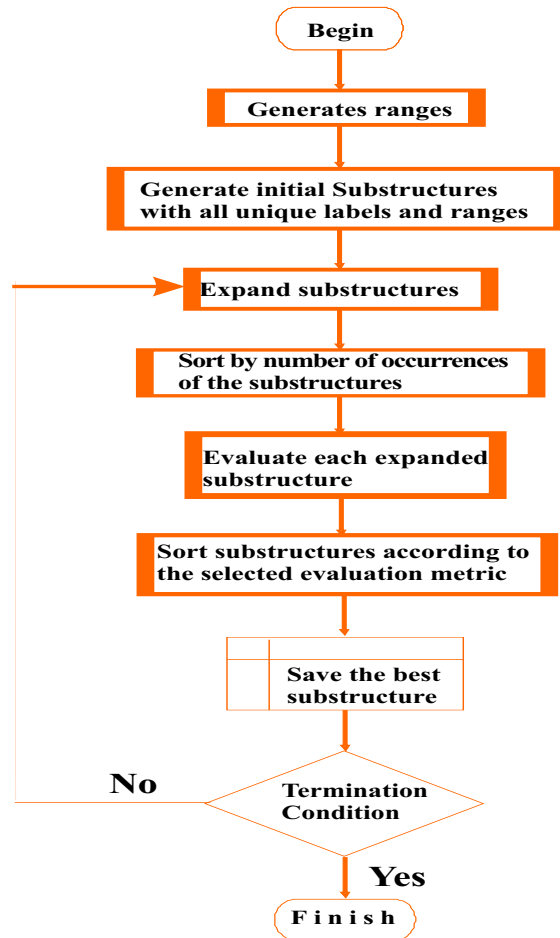


Figure 8. Modification to the Subdue's Process to Search for the Best Substructure using Numerical Attributes.

distance [10], a modification to the Correlation distance [33], a modification to the Camberra distance [36], and a new distance that we propose.

Because of space restrictions, we only show the numerical ranges generation algorithm (using frequency histograms), which can calculate distances using any of the six measures mentioned before.

NumericalRangesHandlin-2(classes, method, attributes, objects, neighbors)

```

listhistogram = null
while classes > 0
  while attribute > 0
    listhistogram = histogramfrequency(attributes, objects)
    ratio = minimum distance between values of the listhistogram
    distance = distanceMethodoour(method, classes, listhistogram)
    threshold = distance * ratio
    for N = 1 to listhistogram
      centroide = listhistogram[N]
      for T = 1 to listhistogram

```

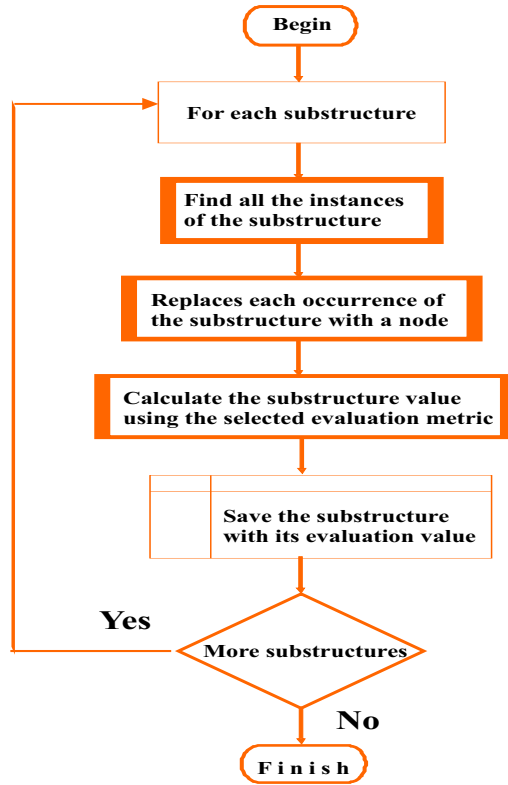


Figure 9. Subdue's Substructure Evaluation Process. This Process will be Modified to work with Numerical Attributes.

```

K = |centroide - listhistogram[T] |
if K < threshold
  newcentroide = (centroide+listhistogram[T])/2
  if newcentroide ∈ listhistogram
    listhistograminstance add + 1
  else
    listhistogram add newcentroide
    listhistograminstance add + 1
K = 0
fusion (listhitogram)
attribute = attribute - 1
classes = classes - 1
  
```

Figure 4.10: Numerical Ranges Generation Algorithm, ("Generated Ranges" from figure 4.8)

The algorithm shown in figure 4.10 works as follows. We First sort the data for each variable (in ascending order for attributes values), for this we use a modification of the radix sort algorithm. We then calculate the histogram of possible values, at worst all the values of this histogram will have frequency of one. We look for the shortest distance between two values, which will be constant K. Distances from the centroid are

calculated to all neighbors based on the standard deviation. The ratio is the multiplication of the standard deviation modified by constant k (shortest distance between two data elements).

The neighboring values were grouped considering a metric based on the vicinity of the numerical values. A new centroid is calculated based on the neighbors added to the range. We incorporate to this range all the neighbors which have a distance shorter or equal to the calculated ratio. We then generate again the histogram of frequencies. The highest peaks of frequency are selected to consider them as centroid (central trend for regions). Based on the neighboring heuristic we calculate the limits of each range of numbers (each range corresponds to a centroid) which is our new range, (this can be seen as a modification of the k -means algorithm).

Since we have six ways to calculate the distance between numerical values, we also have six ways to generate numerical ranges. Using each of these distance measures we can generate different sets of numerical ranges for the same numerical data.

For the evaluation of our work we will compare our work with the C4.5 decision trees algorithm. We will not be able to compare with other graph-based algorithms because those implementations are not available (although we will try again to obtain them). Our preliminary results are presented in next section.

5 Results

The idea of this thesis is to add to the graph based knowledge discovery system Subdue the capacity to handle numerical ranges. The experiments were done with the data representation for numeric attributes (attributes with continuous values) that we are designing.

In this section we present the most significant results for the set of experiments mentioned in the previous section.

Our experiments are oriented to show that by means of the proposed methodology, the objectives of this research work are feasible to reach. These experiments can be divided in two sets, according to the data mining task.

- The first task is related to the concept description process and
- The second task consists of the classification process.

This thesis focuses in the numerical ranges handling for graph based knowledge discovery. For the first test we use a flat domain (a database in the form of a table where each line represents an object and each column represents an attribute or characteristic, by means of which the objects are described) we transformed it into its graph-based representation, then we performed both data mining tasks (concept description and classification) with the Subdue system. We then compared the obtained results with our proposed method for numerical ranges handling in two senses.

- First we tested the use of static numerical ranges, which are generated at the beginning of the data mining process (as a process of discretization) and they continue being used up to the end (until specified number of iterations is reached) for any of the two tasks.
- We then test the use of dynamic numerical ranges, as described in our objective for both data mining tasks. For this experiment we dynamically (but manually at the moment) generate our numerical ranges at each iteration.

In order to show how Subdue's results improve when it is able to deal with numeric attributes we performed our first experiment where we did not give any special treatment to any numeric attribute. With our second

test we want to show that Subdue is able to find interesting patterns containing numeric data when we discretize the numeric attributes as a pre-processing phase. Finally, considering the second test, we performed additional experiments to conclude that the patterns found using discretization in a pre-processing phase are different from those found when incorporating the creation of numerical ranges at the moment of selecting the best patterns (at the end of each iteration). In the case of Subdue, it uses a MDL measure to choose the best substructures. This is what we call dynamic numerical ranges that are created every time that Subdue finds a pattern, thus the substructures found using this technique differ from those found with discretization (during the preprocessing phase).

It is important to remember that these experiments were performed in a semi-automatical way, because we have not implemented them in the Subdue system. We obtained the results of these two experiments (statically and dynamically generated numerical ranges) only for 4 databases.

We used 4 databases with numeric attributes (taken from the UCI Machine Learning Repository [1]), we discretized the numeric attributes and transformed the databases into their graph-based representations to give them as input to the Subdue system. Because of space constraints, we only show our results for the description task for the BUPA database (although we performed the same experiment for the ABALONE, PIMA, and WINE databases), but we show the results of the 4 databases for the classification task.

The database used in our experiments is named “BUPA” and describes objects of liver disorders. The first 5 variables are all blood tests which are thought to be sensitive to liver disorders that might arise from excessive alcohol consumption. Each line in the bupa.data file constitutes a record of a single male individual. It contains two classes **A** and **B** with 345 objects, 145 of which belong to class **B** and 200 that belong to class **A**. Each example is described with 7 attributes or characteristics, of these attributes 6 are continuous, and the last one is nominal (the class). For example, mcv (mean corpuscular volume) are continuous, alkphos (alkaline phosphotase) are continuous, sgpt (alamine aminotransferase) are continuous, sgot (aspartate aminotransferase) are continuous, gammagt (gamma-glutamyl transpeptidase) are continuous, drinks are continuous, and class is nominal. In addition, for this database the classes overlap (the classes have attributes values that overlap), for which the classification process for any algorithm turns out to be more complex.

The results obtained by Subdue without numerical ranges handling for the task of concept description for the BUPA database are shown in figure 10. Subdue was only able to find only a substructure refereing the class but it did not use any numerical attribute. This result is not good for the concept description task. For



Figure 10. Pattern Found by Subdue without Using Numerical Ranges in the BUPA Database (three Substructures in three iterations).

a) First Substructure with 144 instances. b) Second Substructure with 53 instances. c) Third Substructure with 22 instances.

the second test we discretized the numeric attributes during a pre-processing phase. As we can see in the

figure 11 the pattern found (for the BUPA Database) by Subdue contains four vertices related to numerical attributes that could not be found without the pre-processing discretization phase (as shown in the results for the first test).

In our future work we will incorporate the creation of numerical ranges to the patterns search engine so that these can be created dynamically. This will give us more precise results related to patterns found by Subdue in previous iterations and will differ from those patterns found when using discretization as a pre-processing step.

The results obtained for the concept description task with static numerical ranges (for the BUPA Database) are shown in figure 11. As we can see, when Subdue uses static numerical ranges (in a pre-processing discretization) it is able to find larger substructures than when it does not use them (as those shown in figure 10). These substructures are good for the concept description task as they contain a larger number of vertices and some of their labels contain numerical ranges. Then, graph based algorithms are an alternative when we work with mixed data (mainly in structural domains). In the third test (named dynamic numerical ranges) we

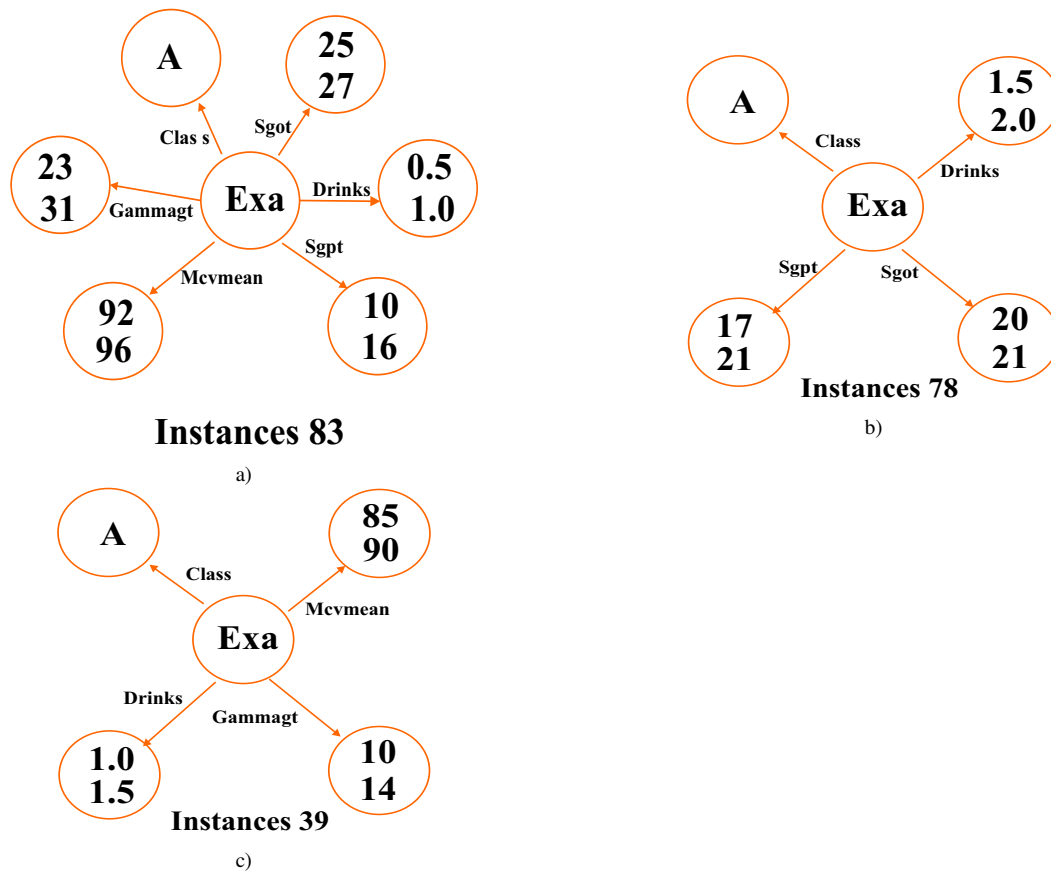


Figure 11. A Pattern Found by Subdue Using Static Numerical Ranges in the BUPA Database (three Substructures in three iterations).
 a) First Substructure with 83 instances. b) Second Substructure with 78 instances. c) Third Substructure with 39 instances.

dynamically (but manually) created our numerical ranges for each iteration of Subdue. The results obtained for the concept description task with dynamic numerical ranges for the BUPA Database are shown in figure 12, in these experiments we found a substructure at each iteration, this substructure was manually replaced

with a vertex that represents it (as Subdue does it internally without the use of numerical ranges), we then used this new input graph for the next iteration of Subdue, and repeated the process until the predefined number of iterations is reached (the stop criterion).

In this case we used 5 iterations, performing in each of them a new regeneration of numerical ranges. In the case of the dynamic numerical ranges generation we replaced the labels of the attributes with a range as in the case of the static numerical ranges regeneration (discretization as a pre-processing phase) but we do it after each iteration because we calculate the new numerical ranges with the vertices that were not compressed by the new found substructure (we actually use the original numeric labels of the vertices to create the new numerical ranges).

We generate new numerical ranges labels that are considered by the candidate's generation process and Subdue then compares labels representing numerical ranges. When we implement the dynamic numerical ranges handling in Subdue we will also need to compare a number with a range or in the simplest case, two numbers (as we mentioned in the objectives of this work). The previous three tests show how Subdue works

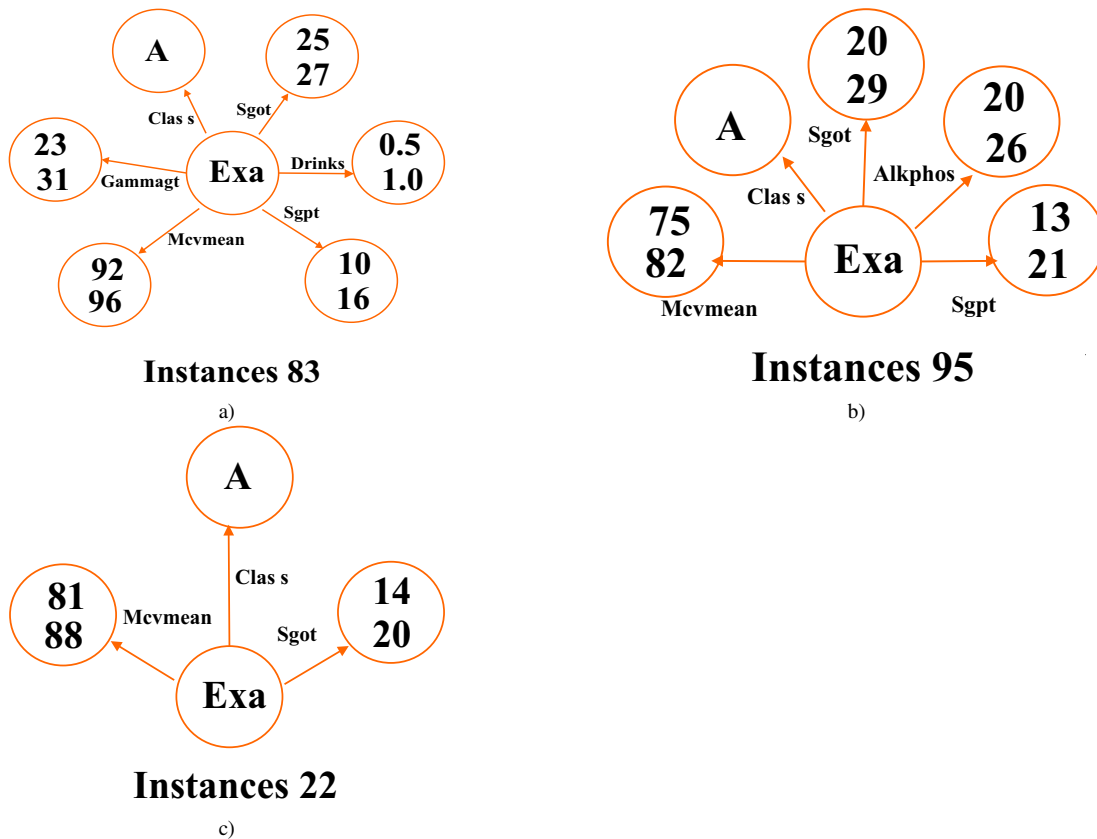


Figure 12. A Pattern Found by Subdue Using Dynamic Numerical Ranges in the BUPA Database (three Substructures in three iterations).
a) First Substructure with 83 instances. b) Second Substructure with 95 instances. c) Third Substructure with 22 instances.

for the concept description task with and without numerical ranges handling and we could show the benefits of using numerical ranges in the last two tests. We also showed that the substructures found with dynamic numerical ranges are different from those found with static numerical ranges. Now we want to show the benefit of this difference for the classification task. We then performed a 3-fold-cross-validation test to verify

how efficient is the model for the classification task. The three-fold cross validation consists of the creation of 3 data sets, two of those data sets are used to train Subdue and the other one for validation. These data sets were randomly created. The experiments for the 3-fold cross validation were done semi-automatically because Subdue does not have an automatic way to do it.

The results obtained by Subdue without using numerical ranges handling for the classification task are shown in table 2, the model obtained by Subdue achieves an accuracy of 45.5% for the BUPA database, 47.7% for ABALONE, 49.6% for PIMA, and 53.4% for the WINE database.

The results obtained for the classification task with static numerical ranges are shown in table 2 63.6% for BUPA, 65.8% for ABALONE, 80.5% for PIMA and 90.2% for WINE. These results show that the classification level improved significantly compared to those without using numerical ranges, and it is also able to find richer models (expressiveness) and larger (in the number of vertices and edges), to describe the set of positive examples. Something important is that the model contains labels with numerical ranges, which is the main point of this work. The model found without numerical ranges is not of great utility because it lacks of meaning.

The results obtained for the classification task with dynamic numerical ranges are also shown in table 2 67.0% for BUPA, 69.0% for ABALONE, 83.1% for PIMA, and 97.3% for WINE. As we described before we ran Subdue for 5 iterations, in each iteration we regenerated our numerical ranges, and next we eliminated the set of training examples covered by the substructure (this was a really time consuming task that was necessary to do in order to perform the dynamic numerical ranges test) and after that, we use the remaining examples to regenerate the new numerical ranges and we repeated this process for 5 iterations. Therefore we obtained 5 substructures that were used for the classification model as we did for the other tests. We used only 5 iterations because this process is extremely time consuming (manually preparing the data for Subdue). In general the use of numerical ranges in the Subdue system for domains with mixed data obtains

Data Base	without Ranges	Static Ranges	Dynamic Ranges	C4.5
BUPA	45.5%	63.6%	67%	62.3%
ABALONE	47.7%	65.8%	69%	52.9%
PIMA	49.6%	80.5%	83.1%	73.4%
WINE	53.4%	90.2%	97.3%	91.8%

Table 2. Accuracy Achieved for the BUPA, ABALONE, PIMA, and WINE Databases.

better results that when it does not use them. These experiments support the extensions to the different algorithms described in this proposal.

All the previous results found by Subdue were obtained without the modification to the comparison heuristic (comparisons between numerical ranges, between numerical labels and between numerical labels and numerical ranges), therefore the numerical ranges could not be compared with numerical labels. This reduces the possibility of using different ranges of numbers (generated with different methods) to expand new candidate substructures and to find instances of those substructures.

6 Conclusion



The different ways to generate numeric ranges used in this work helped us to find significant patterns at different levels of abstraction by means of Subdue.

With our proposal of creating dynamic numerical ranges, we obtained richer descriptive patterns that also obtain a better accuracy for the classification task. This proves that our theory for numeric ranges handling with graph-based representations in graph based knowledge discovery is promising.

The main contribution of this work considers a guideline for the creation of a graph-based representation for mixed data types (continuous and nominal) and also the creation of an algorithm for the manipulation of these graphs with numeric ranges for the data mining task (classification and description).

This will also allow working with domains represented with graphs containing numeric attributes in a more effective way.

7 Chronogram of Activities

The following table shows the global activities programmed by four-month periods for the development of this doctoral investigation. Completed Activities  Activities to be completed 
































































Chronogram of Activities	Four-month period								
	1	2	3	4	5	6	7	8	9
Review of the Literature									
Critical Analysis of the Clustering Algorithms									
Critical Analysis of the Numerical Ranges Handling Algorithms									
Critical Analysis of the Knowledge Discovery in Databases Algorithms									
Algorithms									
Conceptual Development of the Algorithm									
Extend a Sub-Graph-Isomorphism Algorithm									
pre-processing phase (Extend numerical ranges handling)									
processing phase (Extend numerical ranges handling)									
post-processing phase (Extend numerical ranges handling)									
Experiments and Tests of the Algorithm									
Comparison of the Algorithm									
Proposal									
Publications									
Thesis									

Table 3. Chronogram of Activities

References

- [1] A. Asuncion and D.J. Newman. UCI machine learning repository. <http://www.ics.uci.edu/Gmlern/MLRepository.html>, University of California, Irvine, School of Information and Computer Sciences, 2007.
- [2] Andi Baritchi, Diane J. Cook, and Lawrence B. Holder. Discovering structural patterns in telecommunications data. In Proceedings of the Thirteenth International Florida Artificial Intelligence Research Society Conference, pages 82.85. AAAI Press, 2000.
- [3] Ellis J. Clarke and Bruce A. Barton. Entropy and mdl discretization of continuous variables for bayesian belief networks. In International Journal of Intelligent Systems, pages 61.92. Copyright 2000 John Wiley and Sons, Inc., 2000.

- [4] Diane J. Cook and Lawrence B. Holder. Substructure discovery using minimum description length and background knowledge. In *Journal of Artificial Intelligence Research*, volume 1, pages 231.255, 1994.
- [5] Diane J. Cook and Lawrence B. Holder. *Mining Graph Data*. John Wiley and Sons, Inc., John Wiley and sons, inc. edition, 2007. *Mining Graph Data*, Diane J. Cook and Lawrence B. Holder.
- [6] Diane J. Cook, Lawrence B. Holder, and Surnjani Djoko. Knowledge discovery from structural data. In *Journal of Intelligent Information Systems*, volume 5, pages 229.248, 1995. Also available as Technical Report TR-95-149, NASA Center of Excellence in Space Data and Information Sciences.
- [7] Diane J. Cook, Lawrence B. Holder, S. Su, R. Maglothin, and Istvan Jonyer. Structural mining of molecular biology data. In *IEEE Engineering in Medicine and Biology Special Issue on Advance in Genomic*, volume 2, pages 67.74, 2001.
- [8] Sylvain Ferrandiz and Marc Boullé. Multivariate discretization by recursive supervised bipartition of graph. pages 253.264. Springer-Verlag, 2005.
- [9] Scott Fortin. The graph isomorphism problem. Technical Report 1, University of Alberta, Edomonton, Alberta, Canada, 1996. volume 1, pages 20.
- [10] Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramasamy Uthurusamy. From data mining to knowledge discovery: An overview. In *Advances in Knowledge Discovery and Data Mining*, pages 1.34. AAAI Press / The MIT Press, 1996.
- [11] Shayan Ghazizadeh and Sudarshan S. Chawathe. Seus: Structure extraction using summaries. In *DS '02: Proceedings of the 5th International Conference on Discovery Science*, pages 71.85, London, UK, 2002. Springer-Verlag.
- [12] Jesus A. Gonzalez, Lawrence B. Holder, and Diane J. Cook. Structural knowledge discovery used to analyze earthquake activity. In *Proceedings of the Thirteenth International Florida Artificial Intelligence Research Society Conference*, pages 86.90. AAAI Press, 2000.
- [13] Jesus A. Gonzalez, Lawrence B. Holder, and Diane J. Cook. Experimental comparison of graph-based relational concept learning with inductive logic programming systems. In *Lecture Notes in Artificial Intelligence*, volume 2583, pages 84.99. Springer Verlag, 2002.
- [14] Lawrence B. Holder, Diane J. Cook, and Horst Bunke. Fuzzy substructure discovery. In *ML92: Proceedings of the ninth international workshop on Machine learning*, pages 218.223, San Francisco, CA, USA, 1992. Morgan Kaufmann Publishers Inc.
- [15] Lawrence B. Holder, Diane J. Cook, and S. Djoko. Substructure discovery in the subdue system. pages 169.180. Kluwer Academic Publishers, 1994.
- [16] Lawrence B. Holder, Diane J. Cook, Jesus A. Gonzalez, and Istvan Jonyer. Structural pattern recognition in graphs. In *Pattern Recognition and String Matching*, pages 255.280. Kluwer Academic Publishers, 2002.
- [17] Jiawei Han and Micheline Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers, 2nd ed edition, 2006. pages 533. Series in Data Management Systems.

- [18] Akihiro Inokuchi, Takashi Washio, and Hiroshi Motoda. An aprioriBased algorithm for mining frequent substructures from graph data. In *Principles of Data Mining and Knowledge Discovery*, pages 13.23, 2000.
- [19] Akihiro Inokuchi, Takashi Washio, and Hiroshi Motoda. Complete mining of frequent patterns from graphs: Mining graph data. *Machine Learning*, 50(3):321.354, 2003.
- [20] X.Y. Jiang and H. Bunke. Optimal quadratic-time isomorphism of ordered graphs. *PR*, 32(7):1273.1283, July 1999.
- [21] Istvan Jonyer, Diane J. Cook, and Lawrence B. Holder. Concept formation using graph grammars. In *Proceedings of the KDD Workshop on Multi-Relational Data Mining*, volume 2, pages 19.43, Cambridge, MA, USA, 2002. MIT Press.
- [22] Istvan Jonyer, Lawrence B. Holder, and Diane J. Cook. Graph-based hierarchical conceptual clustering. *International Journal on Artificial Intelligence Tools*, 10(1-2):107.135, 2001.
- [23] Istvan Jonyer, Lawrence B. Holder, and Diane J. Cook. Graph-based hierarchical conceptual clustering. *Journal of Machine Learning Research*, 2:19.43, 2001.
- [24] Istvan Jonyer. Context-Free Graph Grammar Induction Using the Minumun Description Length Principle. PhD thesis, The University of Texas at Arlington, 2003. Supervisor-Lawrence B. Holder.
- [25] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31(3):264.323, 1999.
- [26] Michihiro Kuramochi and George Karypis. An efficient algorithm for discovering frequent subgraphs. In *IEEE Transactions on Knowledge and Data Engineering*, volume 16, pages 1038.1051, Piscataway, NJ, USA, 2004. IEEE Educational Activities Department.
- [27] Brendan D. Mckay. Practical graph isomorphism. Technical Report 30, Department of Computer Science, Australian National University, Canberra, ACT 0200, Australia, 1981. volumen 1, pages 20.
- [28] Ivan Olmos Olivera. Common Subgraph Search Based on Vertex-Edge-Vertex Codes and a Breadth-Depth Search. PhD thesis, Santa Maria Tonantzintla, San Pedro Cholula, 2007. Supervisor-Jesus Antonio Gonzalez.
- [29] J. R. Quinlan and R. L. Rivest. Inferring decision trees using the minimum description length principle. In *Inf. Comput.*, volume 80, pages 227-248, Duluth, MN, USA, 1989. Academic Press, Inc.
- [30] Jorma Rissanen. *Stochastic Complexity in Statistical Inquiry Theory*. World Scientific Publishing Co., Inc., River Edge, NJ, USA, 1989.
- [31] Oscar Edgardo Romero Arredondo. Minería de Datos Basada en Grafos Aplicada al Análisis de los Espacios Multifuncinales de las Fincas en Puebla de los A ´ngeles de los Siglos XVI, XVII Y XVIII. Master thesis, Calle Luis Enrique Erro No.1, Colonia Centro, Xalapa de Enriquez Veracruz, 2006. Supervisor-Dr. Manuel Nartinez Morales and Dr. Jesús Antonio González Bernal.
- [32] Singh and Minz. Discretization using clustering and rough set theory. *iccta*, 00:330.336, 2007.

- [33] Shashi Shekhar, Pusheng Zhang, Yan Huang, and Ranga Raju Vatsavai. Chapter 3 Trends in Spatial Data Mining, *Data Mining. Next Generation Challenges and Future Directions*. AAAI/MIT Press, pages 24. Department of Computer Science and Engineering, University of Minnesota, 2003.
- [34] J. R. Ullmann. An algorithm for subgraph isomorphism. *J. ACM*, 23(1):31.42, 1976.
- [35] Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Series in Data Management Systems, second edition edition, 2005. Second Edition (Morgan Kaufmann Series in Data Management Systems) (Paperback), pages 385.
- [36] Ian H. Witten, Eibe Frank, L. Trigg, M. Hall, G. Holmes, and S. Cunningham. Weka: Practical machine learning tools and techniques with java implementations. In *International Workshop: Emerging Knowledge Engineering and Connectionist-Based Info*, pages 192-196. Morgan Kaufmann Publisher, 1999.
- [37] Xifeng Yan and Jiawei Han. gSpan: Graph-based substructure pattern mining. In *ICDM 2002*, pages 51.58, 2002. In *Proc. 2002 Int. Conf. on Data Mining ICDM-02*, Maebashi, Japan.