Handling of Numeric Ranges for Graph-Based Knowledge Discovery

Abstract— **Discovering interesting patterns from structural domains is an important task in many real world domains. In recent years, graph-based approaches have demonstrated to be a straight forward tool to mine structural data. However, not all graph-based knowledge discovery algorithms deal with numerical attributes in the same way. Some of the algorithms discard the numeric attributes during the preprocessing step. Some others treat them as alphanumeric values with an exact matching criterion, with the limitation to work with domains that do not have this type of attribute or discovering patterns without interesting numerical generalizations. Other algorithms work with numerical attributes with some limitations. In this work, we propose a new approach for the numerical attributes handling for graph-based learning algorithms. Our experimental results show how graph-based learning benefits from numerical values handling by increasing accuracy for the classification task and descriptive power of the patterns found (being able to process both nominal and numerical attributes). This new approach was tested with the Subdue system in the Mutagenesis and PTC (The Predictive Toxicology Challenge) domains showing an accuracy increase around 22% compared to Subdue when it does not use our numerical attributes handling method. Our results are also superior to those reported by other authors, around 7% for the Mutagenesis domain and around 17% for the PTC domain.**

I. Introduction

In data mining and machine learning the domain data representation determines in a great measure the quality of the results of the discovery process. Depending on the domain, the Data Mining process analyzes a data collection (such as flat files, log files, relational databases, etc.) to discover patterns, relationships, rules, associations, or useful exceptions to be used for decision making processes and for the prediction of events and/or concept discovery. Graph-based algorithms have been used for years to describe (in a natural way) flat, sequential, and structural domains with acceptable results [1], [2].

Some of these domains contain numeric attributes (attributes with continuous values). Domains containing this type of attributes are not correctly manipulated by graph-based knowledge discovery systems, although they can be appropriately represented. To the best of our knowledge there does not exist a graph based knowledge discovery algorithm that deals with continuous valued attributes in the same way that our new approach does. A solution proposed in the literature to solve 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.

When graph-based knowledge discovery systems first appeared, they were not able to identify that number 2.1 was similar to 2.2, taking them as totally different values. This was the reason why those algorithms did not obtain so rich results as other algorithms that dealt with numeric attributes in a special way (such as the C4.5 classification algorithm that although it does not work with structured domains, it works with numerical data for flat domains). After some years, the number of real world structural domains containing numerical attributes increased as well as the need to have knowledge discovery systems able to deal with that type of attributes. Then, some of the available algorithms were extended in different ways in order to deal with numerical data as we describe in the Related Work Section.

There are two main contributions of this work. The first one consists of the creation of a graph-based representation for mixed data types (continuous and nominal). The second one corresponds to the creation of an algorithm for the manipulation of these graphs with numerical ranges for the data mining task (both, classification and description). In this way, we can work with structural domains represented with graphs containing numeric attributes in a more effective way as we describe in the experimental Results' Section of the paper.

II. Related Work

In this Section, we describe two methods that work with structural domains (and in some way, with numerical attributes) that were used in our experiments. The first is an Inductive Logic Programming (ILP) system: "CProgol" and the second a Graph-based system: "Subdue", which in this work, was extended with our novel method to deal with numerical attributes.

A. Inductive Logic Programming

Logic was one of the first formalisms used in artificial intelligence to represent knowledge structures in computers. Inductive logic programming combines inductive learning methods with the representation power and formalism of first order logic. A logic knowledge representation has the advantage of being sufficiently versatile at covering the needed concepts and at the same time to allow deductive processes from those concepts [3].

ILP is a discipline that investigates the inductive construction of logic programs (first order causal theories) from examples and previous knowledge. Training examples (usually represented by atoms) can be positive (those that belong to the concept to learn) or negative (those that do not belong to the concept). The goal of an ILP system is the generation of a hypothesis that appropriately models the observations through an induction process [3]. ILP systems can be classified in different ways depending on the way in which they generate the hypothesis (i.e. top-down or bottom-up). Some examples of top-down ILP systems are MIS [4] and Foil [5]. Examples of bottom-up ILP systems are Cigol [6], Golem [7], and "CProgol" [8].

The ILP system used in this research to compare our graph-based results is "CProgol". The "CProgol" learning process is incremental. It learns a rule at a time, and it follows the one rule learning strategy. "CProgol" computes the most specific clause covering a seed example that belongs to the hypothesis language. That is, it selects an example to be generalized and finds a consistent clause covering the example. All clauses made redundant by the found clause, including all the examples covered by the new clause, are removed from the theory. The example selection and generalization cycle is repeated until all examples are covered. When constructing hypothesis clauses consistent with the examples, "CProgol" conducts a general-to-specific search in the theta-subsumption lattice of a single clause hypothesis.

Inverse entailment is a procedure that generates a single, most specific clause that, together with the background knowledge, entails the observed data. The search strategy is an A*-like algorithm guided by an approximate compression measure. Each invocation of the search returns a clause, which is guaranteed to maximally compress the data. However, the set of all found hypotheses is not necessarily the most compressive set of clauses for the given examples set. "CProgol" can learn ranges of numbers and functions with numeric data (integer and floating point) by making use of the built-in predicates "is", $<$, $=<$, etc. The hypothesis language of "CProgol" is restricted by mode declarations provided by the user. The mode declarations specify the atoms to be used as head literals or body literals in hypothesis clauses. For each atom, the mode declaration indicates the argument types, and whether an argument is to be instantiated with an input variable, an output variable, or a constant.

Furthermore, a mode declaration bounds the number of alternative solutions for instantiating an atom. Types are defined in the background knowledge through unary predicates, or by CProlog built-in functions. Arbitrary Prolog programs are allowed as background knowledge. Besides the background theory provided by the user, standard primitive predicates are built into "CProgol" and are available as background knowledge.

"CProgol" provides a range of parameters to control the generalization process. One of these parameters specifies the maximum cardinality of hypothesis clauses. Other, defines a depth bound for the theorem prover. One more, sets the maximum layers of new variables. Another parameter specifies an upper bound on the nodes to be explored when searching for a consistent clause. CProgol's Search is guided to maximize the understanding of the theory with a refinement operator, which avoids redundancy. In this way, the search produces a set of high precision rules although it might not cover all the positive examples and might cover some negative examples.

III. Previous work with Subdue

Given a continuous variable, there are different ways to generate numerical ranges from it. The selection of the best set of numerical ranges (obtained with a specific discretization method) is based on its capability to classify the data set with high precision. The evaluation algorithm must also have the property of finding the lower possible number of cut points that generalizes the domain without generating a cut point for each value. The problem now consists of finding the borders of the intervals and the number of intervals (or groups) for each numerical attribute. The number of possible subsets (or ranges) of values for a given attribute is exponential.

"Subdue" had previously dealt with numerical attributes in three different ways (in a limited way). One of them is using a threshold parameter. A second one is applying a-match cost function. The third one is the conceptual clustering version of "Subdue".

In "Subdue" we define the match type to be used for the numerical labels of the input graph. There are three match type conditions. Given two labels "l_i and "l_j, a threshold t (defined as a general parameter): **a) Exact match:** "l_i" $=$ "l_j". **b) Tolerance match:** "l_i matches "$l_j\ iff\ |l_i - l_j| < t$. **c) Difference match:** where a function is matchcost(l_i,l_j) is defined as the probability that "l_j is drawn from a probability distribution with the same mean as l_i and standard deviation defined in the input file [9].

There is another way in which "Subdue" matches instances of a substructure that differ from each other. This is done using the threshold parameter. Note that this threshold differs from the "t" threshold described in [9]. This parameter defines the fraction of the instances of a substructure (or subgraphs) to match (in terms of their number of vertices $+$ edges) that can be different but can still be considered as a match. Here we use a cost function that only considers the difference between the size of the substructure and its instances. The function cost is defined as match-cost(substructure, instance) size(instance) threshold. The default value of the threshold parameter is set to 0.0, which implies that graphs must match exactly.

The conceptual clustering version of "Subdue" (SubdueGL) uses a post-processing phase that joins the substructures that belong to the same cluster. It is based on local values of the variable using the concept of variable discovery [10]. With this approach, SubdueGL finds numerical labels of some of the existent values for each numerical variable. These substructures are then given to "Subdue" as predefined substructures in order to find their instances. The instances of these substructures can appear in different forms throughout the database. Then, an inexact graph match can be used to identify the substructures instances. In this inexact match approach, each distortion of a graph is assigned a cost. A distortion is described in terms of basic transformations such as deletion, insertion, and substitution of vertices and edges (graph edit distance). The distortion costs can be determined by the user to bias the match to particular types of distortions [11]. This implementation allows finding substructures with small variations in their numerical labels. It groups them in a cluster and allows them to grow in a post-processing phase. This post-processing is local to each variable, and therefore, it involves a new search of the substructures. This process consumes extra time and resources (SubdueGL [10]). This happens because the substructures to be explored to find more instances with the new Variables' Representation had already been found. "Subdue" will not discover new knowledge, it will only refine those predefined substructures (clusters).

The previous Subdue's Approaches deal with numerical attributes. However, it is important to choose adequate values for the $matchcost$, $threshold$, and conceptual clustering parameters. These parameters control the amount of inexactness allowed during the search of instances of a Substructure. Consequently, the quality of the patterns found is affected. The problem with discovering patterns in this way is that the user has to make a guess about the amount of threshold that will produce the best results. This is a very complex task.

Our new graph-based approach to deal with numerical attributes differs from others because it considers numerical ranges during the search of the substructures (in the processing phase). We do not use initial information about the domain neither matching parameters. The numerical ranges used at each iteration are dynamic (regenerated at each iteration). This creates better substructures, more useful for the classification task as we show in the experimental Results' Section.

IV. Dealing with Numerical Ranges

Discrete values have important roles in the knowledge discovery process. They present data intervals with more precise and specific representations, easier to use and understand. They are a data representation of higher level than that using continuous values. The discretization process makes the learning task faster and precise.

The main point of a discretization process is to find partitions of the values of an attribute which discriminate among the different classes or groups. The groups are intervals and the evaluation of the partition is based on a commitment, few intervals and strong classes discriminate better. Clustering consists of a division of the data samples in groups based on the similarity of the objects. It is often used to group information without a label.

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 classify the dataset with high precision.

The idea of this work is to add to the graph-based knowledge discovery system, "Subdue", the capacity to handle ranges of numbers [12]. In the following subsection, we describe the new numerical attributes handling algorithm that we created. In order to add "Subdue" this capacity, we propose a graph-based data representation, as we show in figure 1 for temperature: "Temp" with a value of "4.5" and humidity: "Hum" with a value of "3.2". For this example of a flat domain, we use a star-like graph, but we will also work with structural domains. We transform each example into a star-like graph with a center vertex named example: "Exa". We then create a vertex for each of the attribute values of the example and link them to the "Exa" vertex. Those edges are labeled with the name of the attribute. We need to extend this representation to allow the use of numerical ranges as we show in figure 2, where attribute "Temp" has now a value between "3.8" and "9.5" representing the range $[3.8, 9.5]$ and attribute "Hum" has a value between "3.0" and "4.7" for the range $[3.0, 4.7]$. This new data representation is working inside of "Subdue" and is transparent to the user.

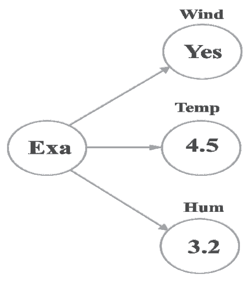

Fig. 1. Labels

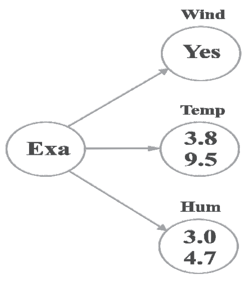

Fig. 2. Ranges Labels

V. Handling of Numerical Ranges

```
GenerateRange (data, N)
  Sort (data)
  for i = 1 to 7
      new histo
      SetInitial (data, N, histo)
      GenerateHistogram (histo)
      distance = TypeofDistance(i, Average(histo), Center(histo), histo)
      threshold = distance + Minimal (histo)
      TypeofGroup(i, threshold, histo)
      rangetable[i] = histo
  return rangetable
```

Fig. 3. Numerical Ranges Generation Algorithm

In this Section, we describe the Numerical Ranges' Generation algorithm (based on frequency histograms). Our algorithm calculates distances using any of seven measures. The first distance that we use is a modification to the Tanimoto distance [13]. The second distance is a modification of the Euclidean distance [15]. The third distance is a modification to the Manhattan distance [16]. The fourth distance is a modification to the Correlation distance [17]. The fifth is a modification to the Canberra distance [14]. The sixth and seventh are two new distance measures that we propose. Figure 3 shows the pseudo-code of our algorithm [12].

The algorithm shown in figure 3, works as follows. The General function (GenerateRange) receives as parameters the dataset and the number of examples. In the first step, and for each numerical attribute, we sort the numerical attribute in ascending way (Sort function). Next, we create a frequency histogram of the ordered data. Then, we create an initial ranges table with four fields corresponding to the center of the range, its frequency, and its low and high limits. In this initial ranges table, the center, the low and high limits contain the same value taken from the Frequencies' Histogram (GenerateHistogram Function). After that, we calculate the minimum distance between any two consecutive rows, using their center fields (Minimal function), and we calculate an average of all the center fields of the frequency histogram (Average function). After that, we calculate the centroid for the center fields of the Ranges' Table, which corresponds to the element of the frequency histogram closest to the value of the average (Center function). We now calculate a type of distance that is used to calculate the grouping threshold (to decide which values are grouped to create a numerical range). In the next step, we calculate a grouping threshold, which is the sum of the minimum distance plus the distance. After that, we iteratively group the elements of the Ranges' Table until the Ranges' Table does not suffer any modification (TypeofGroup function). At this step, we have obtained a final Ranges' Table.

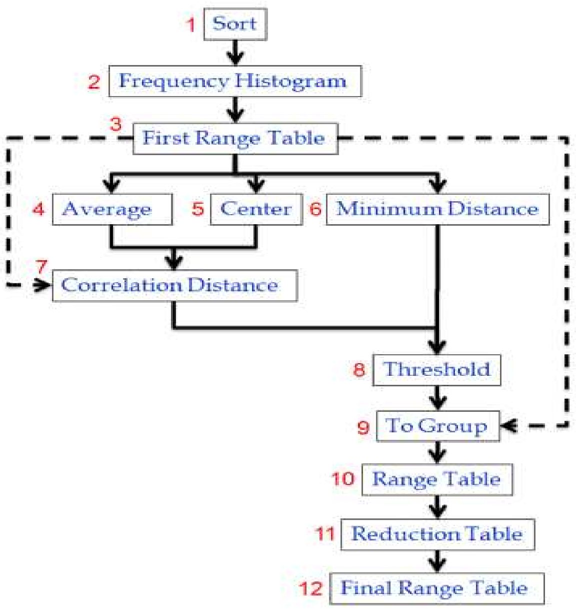

Fig. 4. block diagram

Figure 4 describes our algorithm through a blocks diagram [12]. The blocks diagram shown in figure 4 works as follows. **Block 1.** In this block we sort the values of the numerical attribute using the Quick Sort Algorithm. **Block 2.** In this block we calculate the frequency histogram values, grouping elements with the same value and generating a list composed of two fields, one for the element value and other for its frequency. **Block 3.** In this step we generate the ranges table. This table is composed of four fields, the center of the range, its frequency, and its low and high limits. When we initialize this table, the center and the low and high limits have the same value. The value of the frequency field is taken from the frequencies histogram. Blocks 4, 5, and 6 take as input the center field of the ranges table. **Block 4.** Calculates the average or arithmetic mean of the center field of the ranges table. **Block 5.** Calculates the centroid of all the elements of the center field of the ranges table (numerical attribute), which corresponds to the smaller value closest to the average. (i.e. if the average has the value 6.2 and is the closest value to the average 5.8, then the center field is 5.8). **Block 6.** Calculates the minimum distance between any pair of elements of the center field of the ranges table. (i.e. if element one has a value of 7.1 and element two has a value of 8.4 then the minimum distance is 1.3). **Block 7.** In this step we calculate the distance among the elements of the center field of the ranges table. This calculation depends on the distance type used, in the blocks diagram we use the correlation distance with the formula $R = \frac{\text{cov}(x,y)}{Sx*Sy}$. The distance measures used in this work are modifications of the original equations that we applied to the dataset in order to generate different ranges tables with respect to the number of ranges generated and the number of elements grouped in each range. In order to calculate this distance we use the center (block 5), the average (block 4), and the values of the center field from the ranges table. **Block 8.** Calculates the grouping threshold, which is the sum of the distance type (block 7) plus the minimum distance (block 6). **Block 9.** In this step we perform an iterative grouping process that is repeated until there are no more changes in the ranges table.

1) This grouping is done taking into account two consecutive elements at each time.
2) We first calculate the distance between the pair of consecutive elements and compare this distance with the grouping threshold calculated in block 8 to decide if these center elements should be grouped. The elements are grouped as follows.
3) We create a new range taking as its low limit the lowest limit of both ranges and as its high limit the highest of the two. The frequency is recalculated as the sum of frequencies of both ranges and the new centroid is the average of the low and high limits of the new range.

Block 10. The positive ranges table has been created (the positive ranges table), obtained from the positive examples of the attribute. **Block 11.** The final ranges table, this final ranges table (positive ranges table) is obtained from all the positive examples for this attribute, and in the same way we obtain a final negative ranges table for the negative examples for this attribute (negative ranges table). **Block 12**. This step corresponds to the reduction process of the ranges table considering the SetCovering approach, this grouping process continues until no more changes between the ranges table occur, it is necessary to mention that we only modify the positive ranges table. This reduction is based on the five points of intersection between the ranges.

VI. Structural Databases

In this work, we experimented with different data representations of numerical ranges for two structural databases. The names of the databases are Mutagenesis and PTC (The Predictive Toxicology Challenge for 2000-2001). The National Institute of Environmental Health Sciences (NIEHS) created both databases.

A. Carcinogenesis Domain

The PTC (Predictive Toxicology Challenge) carcinogenesis databases contain data about chemical compounds and the results of laboratory tests made on rodents in order to determine whether chemical compounds induce cancer to them or not. This database was built for a challenge to predict the result of the test using machine learning techniques. The PTC reports the carcinogenicity of several hundred chemical compounds for Male Mice (MM), Female Mice (FM), Male Rats (MR) and Female Rats (FR). According to their carcinogenicity, each of the compounds are labeled with one of the following labels: EE, IS, E, CE, SE, P, NE, N where CE, SR, and P mean that the compound is "relatively active"; NE and N mean that the compound is "relatively inactive", and EE, IS and E indicate that its carcinogenesis "cannot be decided". In order to simplify our problem, we label CE, SE, and P as positive while NE and N are taken to be negative. EE, IS, and E instances were not considered for the classification task.

B. Mutagenesis Domain

The mutagenesis database originally consists of 230 chemical compounds assayed for mutagenesis in Salmonella Typhimurium. From the 230 available compounds, 188 (125 positive, 63 negative) are considered to be learnable (known as regression friendly) and thus are used in the simulations. The other 42 compounds are not usually used for simulations (known as non regression friendly). These databases have been used with different classification algorithms as described in the Results' Section.

VII. Results

In this Section, we present our experimental results with the Mutagenesis and PTC datasets. In the first experiment, we did not give any special treatment to any numerical attribute. In this way, we can show how "Subdue" can be enhanced when adding it the capability to deal with numeric attributes. With our second test, we show that "Subdue" can find interesting patterns containing numerical values using our approach (also improving its classification accuracy). We included our Numerical Ranges' Information to our graph-based representations and executed a 10-fold cross-validation with "Subdue".

A. Graph-Based Data Representation

We generated two general graph-based data representations to be used for both the mutagenesis and PTC databases. The first one considers each compound as a different example. **Compounds** are represented by ***seven attributes*** (**compound element**, **compound name**, **ind1 act** (this value is set to 1 for all compounds containing three or more fused rings), **inda** (this value is set to 1 for the five examples of acenthrylenes as they had lower than expected activity), **logmutag** (log mutagenicity), **logP** (log of the Compound's octanol/water partition coefficient hydrophobicity), and **energy of ϵ LUMO** (energy of the compounds lowest unoccupied molecular obtained from a quantum mechanical molecular model). We also consider that each compound has atoms and links between the atoms and the compound. Each **atom** is represented by ***five attributes*** (**atom element**, **atom name**, **type of atom**, **type of quanta**, and **a partial charge**). Atoms are connected by **bonds** represented by labeled edges. There exist ***eight types of bonds*** (**1=Single**, **2=Double**, **3=Triple**, **4=Aromatic**, **5=Single or Double**, **6=Single or Aromatic**, **7=Double or Aromatic**, and **8=Any**) depending on the type of connection between the atoms. The edge labels for bonds are given by a number, which is not considered to be a numeric attribute since it represents a category.

The graph-based data representation for this dataset is shown in figure 5. The word "Compound" is used to indicate an example in the database. We use this as a central vertex for every example. From this compound vertex, we generate an edge to each of the atoms of the compound, which are labeled as "Element". This compound vertex has also edges and vertices that define the particular characteristics of every compound (attributes of the compound). The edge label gives the name to the attribute, and the vertex label gives it its value. Finally, every atom represented with a vertex with the word "Atom" has edges and vertices to define the particular characteristics of the atom (attributes of the atom). Atoms are connected to other atoms through bond edges, which were described above (seven different types of bonds).

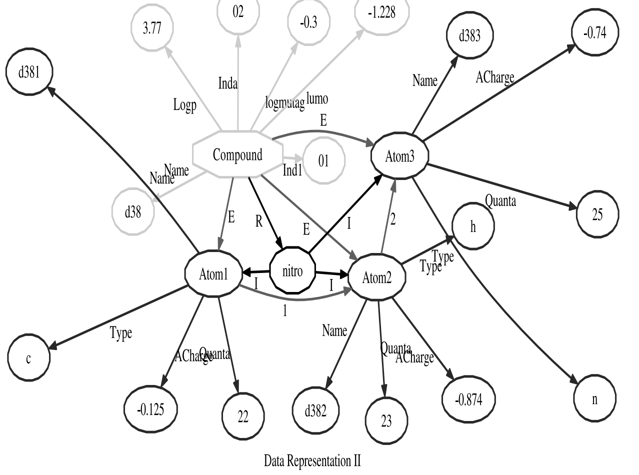

Fig. 5. Partial Graph-Based Data Representation "I".

All the graphs shown in this report were generated with the Graphviz [18] application. This system requires every vertex to have a different name, for example, atom1, atom2, ..., atomN, but this difference does not exist in our graph-based data representation. It is used only for printing purposes. All these vertices have the name "Atom" in our graph-based data representations.

Our second graph-based data representation for the mutagenesis domain is based on the first representation. We added it information about the **aromatic rings** formed by the atoms of the compound in every example. This type of information regards to the **type of ring**. There are ***forty two*** different types of them ("Aromatic", "Ring", "Alkane", "Alkene", "Alkyne", "Amine", "Carbonyl", "Ester", "Carboxylic Acid", "Anhydrid", "Amide", "Alkyl Halide", "Halide", "Aldehyde", "Ketone", "Alcohol", "Thiol", "Ether", "Thio Ether", "Fenol", "Amine Salt", "Imine", "Nitrile", "Iso Cyanate", "Nitro", "Acetale", "Sulfonic Acid", "Sulfonic Amide", "Phosphate Ester", "Phosphor Amidate", "Nitro", "Methyl", "Ring Size 5", "Carbon 5 Aromatic Ring", "Benzene", "Carbon 6 Ring", "Hetero Aromatic 6 Ring", "Ring Size 6", "Anthracene", "Ball 3", "Phenanthrene", "Hetero Aromatic 5 Ring"), and to the atoms that form the ring. In order to represent this data, we add a vertex with the name of the type of ring. We connect one edge to every atom contained in the ring to this vertex. The graph-based data representation can be seen in figure 6.

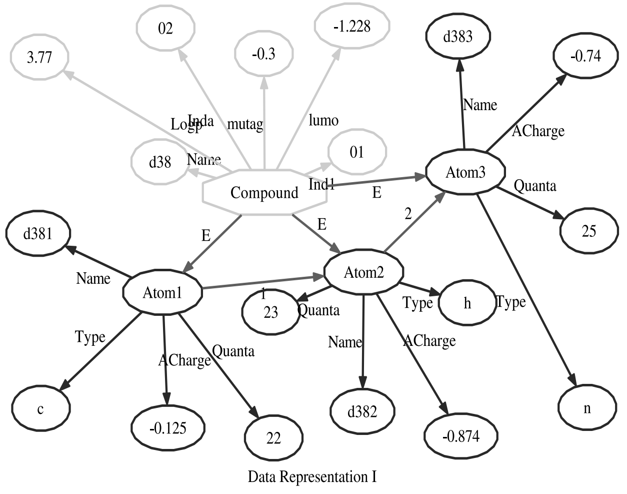

Fig. 6. Partial Graph-Based Data Representation "II".

B. Numerical Ranges Table Generation

The process of Numerical Ranges' Generation creates seven tables of numerical ranges for each numeric attribute. Each table corresponds to a different set of numerical ranges based on the specific type of distance to their neighbors. For this process, we used the reduction of the numerical ranges function. This function creates the numerical ranges of the positive examples in such a way that they do not cover negative examples. For some cases of numerical ranges, all the ranges after the reduction process covered negative examples, being completely eliminated and producing an empty table. In this case, we did not use this type of ranges and only considered the other types of numerical ranges (the positive ranges). We included our Numerical Ranges' Information to our graph-based representations and executed a 10-fold cross-validation with "Subdue". We can see the results for the mutagenesis domain in table I and for the PTC domain in table II.

TABLE I
ACCURACY ACHIEVED FOR THE MUTAGENESIS DATABASE.
10-FOLD-CROSS-VALIDATION.

Type of Graph-Based Data Representation		**Regression**	
		Unfriendly	**Friendly**
Without Rings	Without Ranges	47.23%	58.54%
	With Ranges	86.85%	85.80%
With Rings	Without Ranges	56.12%	61.54%
	With Ranges	81.36%	87.71%

Table I shows the results obtained for the mutagenesis domain with and without the use of numerical ranges for both data representations (with and without rings). The behavior of the results for both representations is stable. We think that by adding rings to representation "I" to create representation "II", we obtained better accuracy results and more descriptive models (with structural information about rings). The input graph of representation "II" is larger than the one created for representation "I". This means that the search space for representation "II" is larger than the one for representation "I". Then, we need to increase Subdue's Parameters in order to find a better model in terms of classification accuracy. However, we obtained a 22% increment when using numerical attributes with both data representations. This means that providing "Subdue" the capability to handle numerical ranges makes it able to find better models.

TABLE II
ACCURACY ACHIEVED FOR THE PTC DATABASE USING A
10-FOLD-CROSS-VALIDATION.

Type of Graph-Based Data Representation		**PTC**			
		MM	**FM**	**MR**	**FR**
Without Rings	Without Ranges	66%	62%	54%	58%
	With Ranges	73%	70%	64%	70%
With Rings	Without Ranges	69%	65%	57%	61%
	With Ranges	78%	74%	72%	83%

Table II shows the results obtained with "Subdue" (with and without ranges) for the PTC domain. In this table, we can see that on average, the classification accuracy increased 17% over the algorithms reported in the literature when we use our proposed method to handle numerical ranges for both data representations (with and without rings). This accuracy increment is not as high as we expected it to be, but as in the previous table, it is due to the execution of "Subdue" with limited parameters. We can also see in the table that the classification accuracy for all the subsets of both domains is stable. This differs from the results reported in related works.

TABLE III
ACCURACY ACHIEVED FOR THE MUTAGENESIS AND PTC DATABASES
USING "CPROGOL" AND A 10-FOLD-CROSS-VALIDATION.

PTC				**Regression**	
MM	**FM**	**MR**	**FR**	**Unfriendly**	**Friendly**
58.93%	55.73%	58.00%	59.10%	67.23%	83.50%

Table III shows the results obtained when we executed

"CProgol" with the second data representation (with rings). The data representations used in "CProgol" are equivalent to those used in our proposed method to handle numerical ranges for graph-based systems ("Subdue"). In these results, we can see that our approach obtains an increase of almost 19.5% for both the PTC and Mutagenesis databases. The results that we obtained with "CProgol" are slightly inferior (around 3% to 4%) than those reported in the citations. This might happen because the background knowledge (or the parameters setting) used in those other works could be different to those used by us. We should also consider that "Subdue" does not use background knowledge, and that we executed "Subdue" with limited parameters. The background knowledge used by "CProgol" consists of a set of rules to describe the Rings' Structures, but we cannot define data types in "Subdue" as it is done in "CProgol". We compared the results of the previous table (the Mutagenesis and PTC databases) with the results of other authors (as shown in the Related Work Section). As we can see, the numerical ranges handling (using numerical and structural data at the same time) that we used with the graph-based data mining system "Subdue", increased Subdue's Accuracy with respect to other algorithms. Analyzing our results we can see that when we add structural data to representation "I" (which uses numerical data and some basic relations between its attributes) to obtain representation "II" (we add relations based on the Rings' Components), accuracy increases by 13% (on average) with respect to the accuracies obtained without using rings.

VIII. Conclusion and Future Work

There are two main contributions of this work. The first one consists of the creation of a graph-based representation for mixed data types (continuous and nominal). The second, the creation of an algorithm for the manipulation of these graphs with numerical ranges for the data mining task (classification and discovery). For our future work we will test different domains to enrich the results of our approach. We will also include temporal information with numerical values. After we have collected this data, we will be able to perform a spatial and temporal data mining process. Finally we will compare our results with "Subdue" against other algorithms that can deal with structural representations with other inductive logic programming systems and we will continue the comparison with "CProgol". This new approach was tested with the "Subdue" system in the Mutagenesis and PTC domains showing an accuracy increase around 22% compared to "Subdue" when it does not use our numerical attributes handling. Our results are also superior to those reported by other authors, around 7% for the Mutagenesis domain and around 17% for the PTC domain. The subdue model helped to distinguish models of Mutagenesis and PTC from others in a better way than "CProgol" (accuracy results are shown in the Results Section) because it is more descriptive. Finally, the substructures found with subdue are richer in structure without need to specify background knowledge to understand them.

Acknowledgment

The first author acknowledges Conacyt for the support provided in his doctoral studies with the scholarship number 86997.

References

[1] 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 ArtificialIntelligence*, volume 2583, 2002, 84-99, (Springer Verlag).

[2] N. S. Ketkar, Lawrence B. Holder and Diane J. Cook, Comparison of graph-based and logic-based multi-relational data mining, *SIGKDD Explor Newsl*, 7(2), 2005, 64-71.

[3] A. Srinivasan, S. H. Muggleton, M. J. E. Sternberg, and R. D. King, Theories for mutagenicity: a study in first-order and feature-based induction *Artificial Intelligence*, volumen 85, 1996, 277-299.

[4] E. Shapiro, Inductive inference of theories from facts, *Computational Logic: Essays in Honor of Alan Robinson*, 1991, 199-255, (Publisher MIT).

[5] J. R. Quinlan, Determinate literals in inductive logic programming, *In IJCAI'91: Proceedings of the 12th international joint conference on Artificial intelligence*, 1991, 746-750, (San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.).

[6] S. Muggleton and W. Buntine, Machine invention of first-order predicates by inverting resolution, *In Proceedings of the 5th International Conference on Machine Learning*, 1988, 339-352, (Ann Arbor, Michgan, USA: CA: Morgan Kaufmann).

[7] S. Muggleton, W. Building and P. Road, Inverse entailment and progol, *New generation Computing*, volume 13, number 3, 245-286, 1995.

[8] S. Muggleton and J. Firth, Cprogol4.4: a tutorial introduction, *In Inductive Logic Programming and Knowledge Discovery in Databases*, 2001, 160-188, (Editorial Springer-Verlag).

[9] A. 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*, 2000, 82-85, (AAAI Press).

[10] I. Jonyer, Lawrence B. Holder and Diane J. Cook, Concept formation using graph grammars, *In Proceedings of the KDD Workshop on Multi-Relational Data Mining*, volume 2, 2002, 19-43, (Cambridge, MA, USA: MIT Press).

[11] I. Jonyer, Lawrence B. Holder and Diane J. Cook, Graphbased hierarchical conceptual clustering, *International Journal on Artificial Intelligence Tools*, 2001, 10(1-2), 107-135.

[12] Oscar E. Romero A., Jesus A. Gonzalez and Lawrence B. Holder Handling of numeric ranges for graph-based knowledge discovery, *In FLAIRS Conference*, 2010.

[13] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*, (Morgan Kaufmann Publishers, 2nd ed edition, Series in Data Management Systems, 2006, pages 533).

[14] I. H. Witten, E. 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*, 1999, 192-196, (Morgan Kaufmann Publisher).

[15] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques. Morgan Kaufmann Series in Data Management Systems*, (Second Edition, Morgan Kaufmann Series in Data Management Systems, Paperback, 2005, pages 385).

[16] U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth and R. Uthurusamy, From data mining to knowledge discovery: An overview, *In Advances in Knowledge Discovery and Data Mining*, 1996, 1-34, (AAAI Press / The MIT Press).

[17] S. Shekhar, P. Zhang, Y. Huang and R. R. Vatsavai, Chapter 3 Trends in Spatial Data Mining, Data Mining, in Editor AAAI/MIT Press (Ed.), *Next Generation Challenges and Future Directions*, (Department of Computer Sciencie and Engineering, University of Minnesota: AAAI/MIT Press, 2003), pages 24.

[18] E. R. Gansner, E. Koutsofios, S. C. North and K. phong Vo, A technique for drawing directed graphs, *IEEE Transactions on Software Engineering*, volumen 19, 1993, 214-230.