

# Rule discovery on related objects

**M. Uludağ**  
**European Bioinformatics Institute**  
**Cambridge, United Kingdom**

**M. R. Tolun**  
**Dept. of Computer Engineering**  
**Çankaya University**  
**Ankara, Turkey**

*Abstract - Many-to-many relations are often observed between real life objects. When many-to-many relations are between objects in the same class the data mining process becomes more complicated than mining objects when there are no such recursive relations. Mining objects related to other objects in the same class requires construction and execution of recursive queries and hence interpretation of the results of recursive queries. Here, we describe Rila, a new relational rule discovery system that was extended for mining objects related to other objects in the same class. Experimental results are provided on Mutagenesis and KDD Cup 2001 data sets.*

## 1.0 Introduction

Current relational database systems utilize many advanced technologies to store and serve structured data, such as indexing and query services and transaction management support. For this reason, a significant amount of today's complex scientific and commercial data is stored in relational database systems.

Scaling graph-mining approaches to very large graphs, which are too big to fit in main memory, is an important challenge [1]. The challenge is to find efficient ways to avoid the requirement of loading complete graph data into the main memory. One approach to address this challenge involves partitioning the graph into smaller graphs that can be processed in parallel [2]. An alternative approach is to store graph data in relational database systems and to have discovery algorithms reading graph data directly from relational database systems. Storing graph data on relational databases makes the advanced technologies utilized by all modern database systems be readily available for the graph data.

Schema of a relational data shows structure of the data. A relational schema is usually represented as a diagram, such as entity-relationship diagrams or UML diagrams. In these diagrams, say schema graph, nodes represent the tables and edges represent the foreign keys between the tables.

Relational discovery systems search the schema graph by following the foreign key links. Supervised relational discovery systems generally start searching from a table where each object being analysed is represented by a single row in this table. This table is called 'target table' in [3] and [4], 'primary table' in [5], 'master relation' in [6], and 'hub table' in [7].

A supervised relational learning algorithm requires one attribute of its input instances to be defined as the target attribute. Unsupervised relational learning algorithms, however, do not treat any particular attribute of the input instances as the target attribute (please see [8] for an excellent discus-

sion on categorization of unsupervised relational learning algorithms). Unsupervised algorithms discover frequent patterns in the relational data independent of a target attribute.

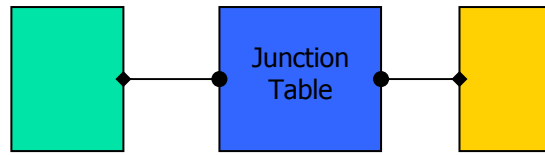


Fig.2. An example many-to-many relation between two different classes

In relational database systems, many-to-many relations are generally defined using junction tables between tables that form the relationship. A junction table contains foreign keys referencing the tables that form the relationship [9]. Querying many-to-many relations is easier if the relation is between objects of two different classes or between two separate groups. Figure 1 pictures an example many-to-many relation where the relation is between the objects in two different classes. However, if there exists a cyclic sub-graph in the relational schema then recursive queries are needed to mine the data. Figure 3 pictures the simplest possible circular structure in a schema where the recursive relation is between objects in the same class.

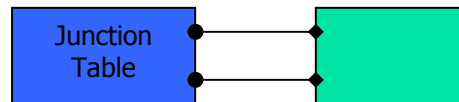


Fig. 1. An example many-to-many relation between objects in the same class

If a many-to-many relation exists between the same group of objects then recursive queries are used for data extraction. Normally all relational data mining systems should be able to handle many-to-many relations between different classes. However, mining many-to-many relations between objects in the same class is more complicated and needs extra care.

In section 2 we will describe the rule discovery system *Rila* that we have used in this study. Section 3 describes a solution for mining many-to-many relations between objects in the same class. In section 4, results of example tests on the mutagenesis and on the KDD Cup 2001 genes data sets are presented. These two data sets both have a many-to-many relation between objects in the same class. For the mutagenesis data we have observed better results than the results known to us previously reported in the literature.

## 2.0 The *Rila* relational rule discovery system

The relational rule discovery system *Rila* that we have improved for mining recursive many-to-many relations was previously described in [10]. The architecture of the rule discovery system developed is shown in Figure 2.

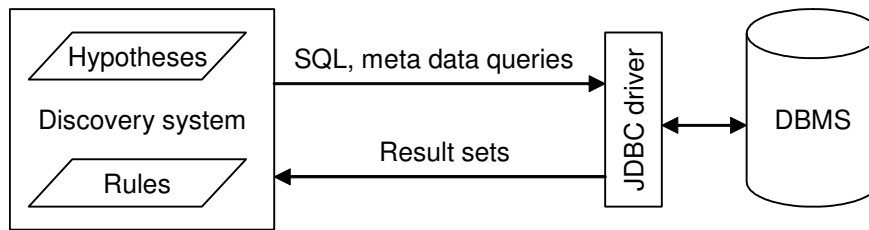


Figure 2. The basic architecture of the *Rila* discovery system

The system was written in Java and uses Java JDBC API to communicate with the database management systems (DBMS). It has a simple graphical user interface through which users can set the mining parameters and start/stop data mining processes. It also has database connection and table selection wizards. After the user selects a set of tables, the target table and the target attribute, a data mining process can be started. When a data mining session is started the system sends meta-data queries to the connected DBMS, to retrieve column and foreign/primary keys information for the selected tables. After the initialization phase, SQL queries sent to the database management system are generally for building valid hypotheses about the data. Basically, the system sends SQL queries to the database system and by analyzing the results of these queries it generates new hypotheses and then new SQL queries to further analyze the data.

*Rila* builds hypotheses in a *levelwise* manner. In each level, it searches the schema graph using a breadth-first search strategy. In each table visited, the system performs two queries for the attribute columns in the table. One query is for collecting the frequency of the positive attribute values and the other query is to learn the frequency of the negative attribute values.

In order to avoid redundant rule selection, *Rila* use the 'effective cover' concept. When a new rule is asserted all the examples covered by the rule are marked in a temporary table, using the primary key in the target table as an identifier for each example. An example cannot be marked by more than one rule. Effective cover of a rule is defined as the number of examples it marks. If a rule's effective cover is zero then it indicates that examples covered by the rule are already covered by the rules previously generated, therefore in this case the rule is not appended to the final rule list.

### 3.0 Many-to-many relations between objects in the same class

In order to better understand the importance of mining many-to-many relations between objects in the same class we can compare the following two example rules:

**Rule 1:** If a gene has a property  $p1$  and interaction of type  $t1$  to any gene then its class is  $c1$

**Rule 2:** If a gene has a property  $p1$  and interaction of type  $t1$  to any gene having property  $p2$  then its class is  $c2$

In this example the interaction relation is a kind of many-to-many relation between genes. If a relational mining system is not fully aware of the recursive many-to-many relations then the rules it produces can't state anything about the attributes of the genes in the other side of the relation, such as in the case of Rule 1. However, a mining system which is fully aware of recursive many-to-many relations can produce rules that can have conditions for the genes in both sides of the relation, such as in Rule 2, for example.

In order to extend the *Rila* for mining many-to-many relations between objects in the same class, we have first modified the search algorithm so that a table in the schema graph could be visited for the second time if it can be reached via an alternative route where the search starts from the target table. Next when a table is visited for the second time, it is handled in a different way using recursive constructs.

In the extension levels of the search process, if a hypothesis includes a condition from a junction table column then it is extended only by the attribute columns on the other side of the relation and/or by the unused attribute columns in the junction table. In order to limit the level of recursion by two, the search algorithm stops when a junction table is visited for a second time. In future work we would like to investigate higher levels of recursion.

In Figure 3 shown is the internal representation of a relational rule, which refers to gene objects related to each other by the interaction relation. The nodes in the horizontal rectangle represent the linked list data structure that stores the conditions in the rule. In addition, each node in this linked list of conditions has a linked list of associations (shown inside the vertical rectangles in the figure) that shows how the condition is related to the objects being referred by the rule. The path of associations in each list goes until it reaches to the target table where each object is represented by a primary key. Considering this point we can imagine each relational rule being a closed and connected graph. This also proves that relational learning is a type of graph mining, i.e. a type of frequent sub-graph mining.

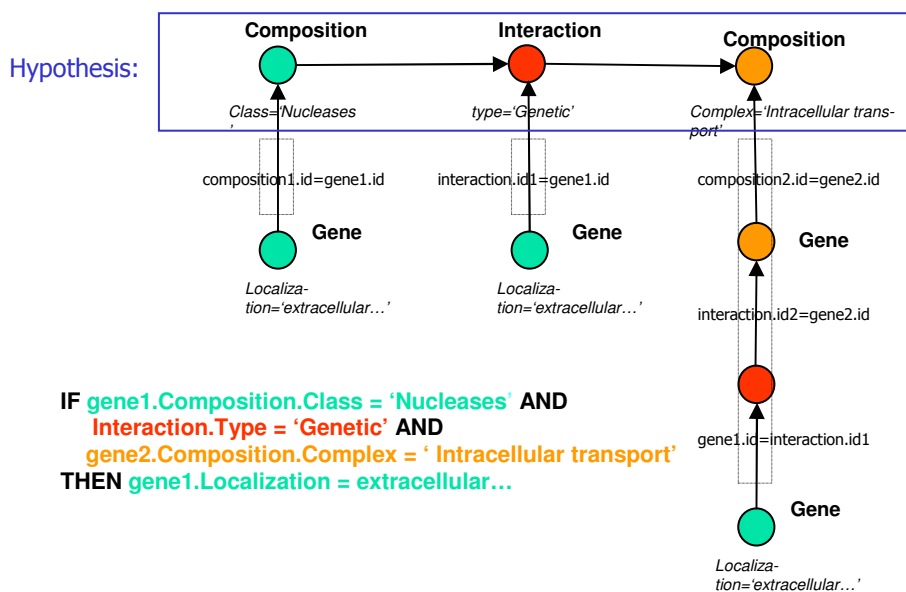


Figure 3. Internal representation of an example relational rule

During the search process, hypotheses having a condition from a junction table column are marked to differentiate them from the single object hypotheses. This mark is later used while generating SQL statements corresponding to the hypothesis. The solution implemented for mining recursive relations use table aliases to construct recursive queries. For example, consider the following hypothesis:

```
INTERACTION.EXPR = (0.026-0.513] AND COMPOSITION.PHENOTYPE =
Auxotrophies
```

In order to extend the above hypothesis by the CHROMOSOME attribute from the GENE table the rule discovery system generates the following SQL statement:

```
Select GENE_B.CHROMOSOME, count (distinct GENE.GENEID) from
COMPOSITION, GENE, GENE GENE_B, INTERACTION where
INTERACTION.GENEID2=GENE_B.GENEID and
INTERACTION.GENEID1=GENE.GENEID and
INTERACTION.EXPR > 0.026 and
```

```

INTERACTION.EXPR <= 0.513 and COMPOSITION.PHENOTYPE = 'Auxotrophies' and COMPOSITION.GENEID=GENE.GENEID and
GENE.LOCALIZATION = 'ER'
group by GENE_B.CHROMOSOME

```

Using the results of the above SQL query, the *Rila* generates new hypotheses that relate genes to each other. In this example, the relation is the gene expression level between the genes and is described by the *EXPR* column of the *INTERACTION* table.

## 4.0 Experiments

We have conducted experiments on two data sets - the mutagenesis database that has been widely used in many areas of data mining research, and the genes database from KDD Cup 2001 competition. We have used Weka machine learning libraries for discretizing all the numeric attributes [11]. We have selected the class blind binning approach, with maximum 15 bins, and the optimize bins option. For the optimistic estimate pruning, the minimum acceptable F-measure value,  $f$ , was set to 0.05 and in each level only the best 20 hypotheses were extended.

### 4.1 Mutagenesis Data Set

The mutagenesis data set consists of 230 molecules divided into two subsets: 188 molecules for which linear regression yields good results and 42 molecules that are regression-unfriendly [12]. This database contains descriptions of molecules. The characteristic to be predicted is their mutagenic activity represented by the attribute *label* in the table *molecule*.

This dataset comes with different levels of background knowledge  $B_0$ ,  $B_1$ ,  $B_2$ , and  $B_3$ . In our experiments we chose to use the background knowledge  $B_2$  and regression friendly subset of the dataset. We split the dataset of 188 compounds into 10 cross-validation sets for testing. We have repeated several tests using different parameters; the best average accuracy that have been observed using ten-fold cross-validation was 90.9% when the majority class selected as default rule and applied to the instances not covered by other rules. When we didn't have a default rule the best accuracy observed was 94.2% and 171 of the 188 compounds (90.96%) were covered by the generated rules. This is better than the results recently reported in [13] which is 87.5%. It is also better than the best results reported by the originators [12] of the data set which is 89.4% where accuracy is defined as number of correct predictions divided by the number of predictions made for all drugs predicted.

Table 1. Cross validation accuracy of the generated rules for the mutagenesis data set.

	Bond table not included	Bond table included, recursive mining
No default rule	94.2% (173 examples covered)	93.1% (175 examples covered)
Majority class as the default rule	90.9%	89.9%

## 4.2 KDD Cup 2001 Data Set

Tests also conducted on the KDD Cup 2001 ‘genes’ data [14]. The database consists of 3 tables, GENE, INTERACTION and COMPOSITION, where GENE table contains 862 entries in the training set and 381 in the testing set. The interaction table defines the relation between the genes.

Tests were conducted to generate classifiers for the ‘gene’ localization attribute using the following settings: maximum number of rule conditions was limited to 5; the penalty factor parameter was set to 2; unknown values were ignored; function column in the composition table was not included in the analysis.

Table 2. Test results on the genes data for predicting the localization attribute.

	Using the interaction table	Interaction table was not included
Number of rules	79	50
Number of conditions	326	152
Training set coverage (%)	61.7	54.76
Test set accuracy (%)	82.7	86.6
Test set coverage (%)	62.2	54.8

The results shown on Table 2 indicate that rules found by recursive mining have better accuracy on the test than the rules found when the interaction table excluded. However when the interaction table excluded we observe more examples covered in the test set. This is related to the size of the rules, as the number of conditions increase the generalisation capacity of a rule decreases.

## 5.0 Discussion and Conclusions

In order to make better use of relational data, a relational discovery system should be able to mine recursive structures. We have developed a solution for mining recursive structures in a relational database and implemented this solution on the *Rila* relational learning system. The datasets used and the implementation of the algorithm are available in the project web site at <http://cmpe.emu.edu.tr/rila/>.

The motivation behind developing a solution for mining recursive relations was the expectation of producing prediction models that may have better classification efficiency. We have performed tests on two example data sets by including and by excluding the recursive many-to-many relations they have. Although the system produce different rule sets in two different cases, we could not observe example instances that recursive mining produces clearly better prediction performance. We have found that the organizers of the KDD Cup 2001 competition had similar expectations when they included the interaction table as part of the genes data set [14]. However, they were surprised when the interaction information was only of some value but not particularly great value in one of the winning approaches and it was the least important item in another winning approach in the competition. This might be related to the fact that the data sets are missing too much interaction information for it to be highly useful. Another reason should be related to the fact that rules generated by recursive mining have more conditions than the other case therefore they have naturally less generalisation capacity. The rules found by recursive mining may not always be successful for prediction but they are good in explaining the nature of a recursive relation.

## 6.0 References

- [1] Holder L. and Cook D., Graph-based Relational Data Mining: Current and Future Directions, SIGKDD Explorations Special Issue on Multi-Relational Data Mining, Volume 5, Issue 1, 2003.
- [2] Cook D., Holder L., Galal G., and Maglothin R. Approaches to parallel graph-based knowledge discovery. *Journal of Parallel and Distributed Computing*, 61(3):427-446, 2001.
- [3] Knobbe, A.J., Blockeel, H., Siebes, A., Van der Wallen, D.M.G.: Multi-Relational Data Mining, In *Proceedings of Benelearn'99*, (1999)
- [4] Leiva, H., and Honavar, V.: Experiments with MRDTL—A Multi-Relational Decision Tree Learning Algorithm. In *Dzeroski, S., Raedt, L.D., and Wrobel, S. (editors): Proceedings of the Workshop on Multi-Relational Data Mining (MRDM-2002)*, University of Alberta, Edmonton, Canada, (2002) 97-112
- [5] Crestana-Jensen, V. and Soparkar, N.: Frequent Item-set Counting across Multiple Tables. *PAKDD 2000*, (2000) 49-61
- [6] Wrobel, S.: An Algorithm for Multi-Relational Discovery of Subgroups, *Proceedings of Principles of Data Mining and Knowledge Discovery-97*, Springer-Verlag, Berlin, New York, 1997.
- [7] SRS-Relational White Paper, Working With Relational Databases using SRS, LION Bioscience Ltd. <http://www.lionbioscience.com/solutions/products/srs>
- [8] Blau, H. and A. McGovern (2003). Categorizing unsupervised relational learning algorithms. *Proceedings of the Workshop on Learning Statistical Models from Relational Data, Eighteenth International Joint Conference on Artificial Intelligence.*
- [9] Microsoft SQL server online documentation
- [10] Uludag, M., Tolun, M.R., Etzold, T.: A Multi-Relational Rule Discovery System, *Proceedings of Eighteenth International Symposium on Computer and Information Sciences*, Antalya, Turkey, November 2003.
- [11] Witten, I.H., Frank, E.: *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*, Morgan Kaufmann Publishers, San Francisco, California, 2000.
- [12] King R.D., Muggleton S.H., Srinivasan A. and Sternberg M.J.E., Structure-Activity Relationships Derived by Machine Learning: The Use of Atoms and their Bond Connectivities to Predict Mutagenicity by Inductive Logic Programming, *Proc Natl Acad Sci U S A*. 1996 January 9; 93 (1): 438–442.
- [13] Atramentov A., Leiva, H., and Honavar, V. A Multi-Relational Decision Tree Learning Algorithm: Implementation and Experiments. In: *Proceedings of the Thirteenth International Conference on Inductive Logic Programming*. Berlin: Springer-Verlag. In press. (2003)
- [14] Cheng, J., Krogel, M., Sese, J., Hatsiz, C., Morishita, S., Hayashi, H., and Page, D.: KDD Cup 2001 Report, ACM Special Interest Group on Knowledge Discovery and Data Mining (SIGKDD) Explorations, vol. 3, issue 2, (January 2002) 47-64.