# Knowledge Discovery with Genetic Programming for Providing Feedback to Courseware Authors

CRISTÓBAL ROMERO*, SEBASTIÁN VENTURA* and PAUL DE BRA†
**Dept. of Computer Sciences and Numerical Analysis. University of Córdoba. 14071 Córdoba (Spain). E-mail: cromero,sventura@uco.es*

*†Dept. of Computer Sciences. Eindhoven University of Tecchnology. PO Box 513, 5600 MB Eindhoven (Netherlands). E-mail:debra@win.tue.nl*

July 1, 2004

**Abstract.** This paper describes how to use evolutionary algorithms as data mining methods for discovering prediction rules in databases. These rules will be used to improve courseware, especially Adaptive Systems for Web-based Education (ASWE). Our aim is to discover important dependence relationships among the usage data picked up during students' sessions. This knowledge may be very useful for the teacher or the author of the course, who could decide what modifications will be the most appropriate to improve the effectiveness of the course. In order to perform the discovery of rules we have used Grammar-Based Genetic Programming (GBGP) with multi-objective optimization techniques. Using that, we can interactively specify the form of the rules and we can choose several evaluation measures of the rules' quality at the same time. Finally, we have developed a specific mining tool for discovering rules which facilitates the usage of the proposed discovering and improving ASWE methodology.[1]

**Keywords:** adaptive system for web-based education, data mining, evolutionary algorithms, grammar-based genetic programming, prediction rules

## 1. Introduction

Web-based Education (WBE) is an important research area at present (Brusilovsky, 2001) due to its beneficial features, such as the physical location independence of students and teachers, and the independence of the computer platform used (PC, MAC, UNIX, etc.). The first developed educational systems were only a net of static hypertext web pages. This led to navigation and comprehension problems because such a net provides uncontrolled navigation freedom. In order to solve this problem, people have started to increase these systems' adaptability and adaptivity (Brusilovsky, 1998). The systems have recently been referred to as Adaptive Systems for Web-based Education (ASWE).

[1] This paper (or a similar version) is not currently under review by a journal or conference, nor will it be submitted to such within the next three months.

These systems are the result of a joint evolution of Intelligent Tutoring Systems (ITS) and Adaptive Hypermedia Systems (AHS) and they have the most desirable characteristics of both: increase of the user interaction and adaptation to the needs of each student.

The development of an ASWE is a laborious activity (Hérin et al., 2002), and it is more complex when the number of adaptation possibilities is higher. The developer, usually the course teacher, has to choose the contents that will be shown, decide on the structure of the contents and determine which are the most appropriate content elements for each type of potential user of the course. So, a careful design is not enough but, in most cases, it is necessary to carry out an evaluation of the system based on students' usage information. It would be also desirable that this evaluation activity was carried out in a continuous way (Ortigosa and Carro, 2002). In order to do that it is necessary to use tools and methodologies able to observe the student's behavior and to attend teachers in the process of continuous improvement of ASWEs, detecting possible errors, shortcomings or improvements that could be carried out.

During the past years, different methods of knowledge discovery or data mining (Zaïne and Luo, 2001) have begun to be used to help teachers in the validation of ASWEs. These methods allow to discover new knowledge starting from the students usage data. This idea has already been successfully applied in e-commerce systems, in which its use is very popular (Spiliopoulou, 2000). However, in the area of e-learning very little research has been done in this direction and this is the research line to be introduced in this paper.

Data mining (DM) is a multidisciplinary area in which several computation paradigms converge: decision tree construction, rule induction, artificial neural networks, instance-based learning, bayesian learning, logic programming, statistics algorithms, etc. (Klösgen and Zytkow, 2002). The objective of data mining is to discover new interesting and useful knowledge. Evolutionary Algorithms (EA) (Freitas, 2002) are one of the new methods applied to do it. The main advantages in the use of EAs over the classical greedy induction algorithms are their ability to do a global search and the way they treat the attribute interaction problem. In the context of rule discovery using EAs (Freitas, 1997), an individual will represent a rule or a group of candidate rules. The fitness function will correspond to some rule quality evaluation measures. A selection procedure will use the fitness value to choose the best rules. Genetic operators will transform the candidate rules into new ones. So, evolutionary algorithms will perform a search in the space of the candidate rules in a similar way to what induction algorithms would do. The difference is in the search strategy used by each one. Thus, classical

learning induction algorithms normally use a local greedy search strategy, while evolutionary algorithms use a global search strategy inspired by natural selection (Darwin, 1859). There are different approaches to evolutionary computing. Specifically, we are going to adopt data mining using grammar based genetic programming (Wong and Leung, 2000).

The knowledge discovered by a data mining method is always desired to be interesting, novel and useful to the end-user (Bayardo and Agrawal, 1999). So, only a subset of interesting rules should be extracted from all the discovered rules. And although data mining algorithms usually discover comprehensible and exact rules, these are often not interesting rules due to the fact that interest is a more ambitious and difficult objective. There are two methods to select interesting rules, namely subjective and objective methods (Liu et al., 2000). The first ones are domain dependent and user directed. On the other hand the objective methods are data directed, domain independent and they use some evaluation measure. In this sense there are a great number of evaluation measures (confidence, support, interest, gini, laplace, etc.) described in literature (Tan et al., 2002) (Lavrac et al., 1999) (Yao and Zhong, 1999) that we have summarized in Appendix A: Rule Evaluation Measures. But each of these measures is centered on a specific aspect of the discovered rules' quality and there is no measure that is significantly better than the other ones in every application domain. For this reason, it can be necessary to consider several of these measures simultaneously. A simple way of doing this is to use a combined metric that ponders by means of weights over each of the measures we use. However, this isn't a good approach because the used measures can be in conflict with each other and can be non commensurable, in the sense that they evaluate very different aspects of the rule. This problem suggests the use of a multi-objective approach (Freitas, 2002) for rule discovery. In this case, the fitness value to be optimized isn't a unique value, but a vector of values, where each value measures a different aspect of the rule quality. Although there is a lot of literature about Multi-Objective Evolutionary Algorithms (MOEA) (Coello et al., 2002) (Deb, 2001), the use of MOEA for discovering rules seems to be relatively unexplored (Ghosh and Nath, 2004).

We have divided this paper in the following sections. First we review the background or research related to the use of data mining in adaptive educational systems. Secondly we describe the specific knowledge discovery problem we want to resolve and the methodology we are going to propose to do it. We show students' usage information and its preprocessing. Next we present the evolutionary algorithms for rule discovery that we have developed using multi-objective genetic programming. After that we present the experimental results of the

different accomplished tests. We then describe the discovered information in the form of rule prediction and its use to improve ASWEs. Finally we draw conclusions to indicate points for further research.

## 2. Related Work

The current necessity to analyze the great quantity of information generated daily by web-based applications, has spawned the interest in using web data mining. Web data mining is the application of data mining methods to web data (Srivastava et al., 2000). There are three different types of web mining depending on the data we use:

**Web Content Mining.** It tries to discover useful information from web content such as metadata, documents, texts, audio, video, hyperlinks, etc.

**Web Structure Mining.** It mines the structure of web hyperlinks, that represent web site hyperlinks graph.

**Web Usage Mining.** It uses data generated by user interaction such as access data (log files), user profiles, transactions, queries, bookmarks, clicks, etc.

The major application areas for web usage mining are (Srivastava et al., 2000): personalization, business intelligence, usage characterization, systems improvement, site modifications, etc. The most developed applications are e-commerce systems. These specific applications try to understand clients' tastes with the objective of increasing web-based sales (Spiliopoulou, 2000).

A more recent application is personalization (Pierrakos et al., 2003), which adapts information systems to users' needs. One case of personalization systems is the use of web mining in educational systems. Using data mining methods in e-learning systems can provide useful information to evaluate and to improve them. Although this research area is still in its infancy, web-based educational systems that exploit the advantages of knowledge acquisition are already being developed (Zaïne and Luo, 2001).

Although discovery methods used in both areas (e-commerce and e-learning) are similar, the objectives are different depending on the point of view. From a system's point of view, there are no differences between them. Since the objective of applying web mining in both application areas is to study clients' behavior (referring to both clients in e-commerce systems and students in e-learning systems), evaluate

this behavior, and improve the systems to help them. But from a user's point of view, there are differences, because the e-commerce objective is to guide clients in purchasing, and the e-learning objective is to guide students in learning. So, each of them has special characteristics that require a specific treatment of the web-mining problem.

Web mining use in education is not new. Some developments have already been carried out with success, applying data mining in web-based educational systems. Usually, their applications consist of searching browsing patterns using one or several of the following algorithms:

- *Association rule mining and sequential pattern analysis.* It consists in searching associations among web pages visited, the first, and analyzing sequences of pages hit in a visit or between visits by the same user, the second. One of the pioneers in this area are Omar Zaïne et al. (Zaïne and Luo, 2001). They propose discovering useful patterns based on restrictions to help educators to evaluate students' activities in web courses. They also use recommender agents for e-learning systems using association rule mining (Zaïne, 2002). The objective is to discover associations between user actions and URLs. In other research (Yu et. al, 2001) they use data mining technology to find incorrect student behavior. They modify traditional web log records that are named learners' portfolios. They apply fuzzy association rules to find relationships among the items with linguistic values like time spent on-line, numbers of articles read, number of articles published, number of questions asked, etc. A related research has been carried out by Pahl and Donnellan (Pahl and Donnellan, 2002). Their work is based on analyzing each student's individual sessions. They define the learning period (of time) of each student. After they split web server log files in individual sessions they apply session statistics, session patterns and session time series. Finally, another work is the one carried out by Wang (Wang, 2002), which uses associative material clusters and sequences among them. This knowledge allows teachers to study the dynamic browsing structure and to identify some interesting or unexpected learning patterns. To do this, he discovers two types of relations: association relations and sequence relations between documents.

- *Clustering and classification.* This consists of grouping users by navigation behavior, grouping similar navigation behaviors, etc. This approach is developed by Tang and McCalla (Tang and McCalla, 2002). They use data clustering for web learning to promote group-based collaborative learning and to provide incremental learner diagnosis. Data clustering finds clusters of students with

similar learning characteristic based on the sequence of pages and content of each page they have visited. Finally, a specific work of using evolutionary algorithms for data mining optimization in educational web-based system is the one carried out by Minaei-Bidgoli and Punch (Minaei-Bidgoli and Punch, 2003). In this work, they want to classify students in order to predict their final grade based on features extracted from the logged data. They use genetic algorithms to optimize a combination of multiple classifiers by weighting feature vectors.

In all the mentioned cases, the current approaches use the visited pages as input data of the search, and so, the result is to discover different types of relations between them. On the other hand, our proposal consists of searching relations between concept and chapter entities of web-based courses, and not between pages. Our objective is to discover relations to restructure not only the browsing paths but also the curricula and contents of adaptive web-based courses.

## 3. Problem Description and Proposed Solution

The task of designing and developing an adaptive system for web-based education is arduous and laborious (Carro et al., 1999) due to the fact that the author of the courseware has to take important decisions about:

- How many chapters or lessons a course consists of and which ones are the most appropriate for each different knowledge level of students. And what is the most adequate organization (navigation scheme) of these chapters.

- How many concepts a chapter consists of and which ones are the most appropriate for each student's knowledge level. And what is the most adequate organization of these concepts.

- Which activities will be used to evaluate each concept and each chapter. And what is the most adequate organization of these activities.

Due to all these decisions, it is very difficult to establish the best course structure, contents and activities. In fact, it is very likely that several authors would propose different curricula for the same course. And also there is an additional problem in this type of systems: the subjectivity when dividing the course into different difficulty levels or

accessibility levels in order to perform a better adaptation or personalization. All the above-mentioned issues show that it isn't easy to develop an adaptive system for web-based education. So, we should improve ASWEs after their construction by means of the evaluation or validation of the system.

The evaluation of an educational system like ASWEs is a process of data gathering to determine the value of the instruction. There are mainly three evaluation paradigms (Arruabarrena et al., 2002): Formative Evaluation, Summative Evaluation and Integrative Evaluation. And there are three kinds of evaluation: Internal Evaluation, External Evaluation and Global Evaluation. Finally, there are specific evaluation techniques for ASWEs. Some of these are: (a) Comparison to evaluate the characteristics of the system versus some standard ones or other systems. (b) Contact with users to collect data about the user interaction, behavior and attitude. (c) Data analysis for reviewing, studying and assessing collection of data about certain characteristics. (d) Pilot testing for studying the performance of the system with potential end-users.

But the problem is that normally teachers only evaluate from a student's point of view and they are centered on evaluating the students' learning based on the scores obtained by them. So, the system is not usually modified after its publications on the web. And if some modifications are done they are based only on teacher judgement or on basic statistical information, for example, questions that more students have failed. But a deeper evaluation of the students' interaction should be carried out.

Our specific evaluation proposal is based on the use of data mining methods on the students' usage data. The originality of this work is the type of knowledge we wish to discover. The related research (Zaïne and Luo, 2001) (Pahl and Donnellan, 2002) (Wang, 2002) (Tang and McCalla, 2002) we have previously described, carries out the discovery of associations between visited pages, analyzing sequences of pages, grouping together browsing patterns or students, etc. But our aim is different since we are going to discover dependence relationships among elements and not among pages. These elements can be concepts, chapter, questions, activities, etc. that are contained in one or several different web pages. We propose a development methodology that enables us to evaluate and improve ASWEs. This methodology is recurrent and the higher the number of students using the system becomes, the more information becomes available to the teacher to improve the course (Figure 1). In this methodology we have added a specific evaluation step using data mining techniques.

The proposed methodology consists of four main steps:

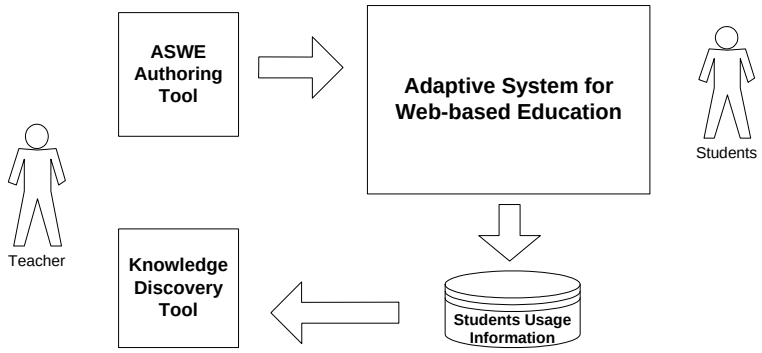


*Figure 1.* Proposed methodology to improve ASWEs

1. **Development of the ASWE.** This is the first stage of the methodology, in which the teacher builds the Hypermedia Adaptive Course providing information about the domain model, the pedagogic model and the interface module. To facilitate this task he normally uses an authoring tool, either of a commercial general-purpose type such as DreamWeaver, Toolbook, Director, etc. or of specific types such as AHA! (De Bra et. al., 2003), HamWeb (De Castro and Romero, 2002), etc. The remaining information: the tutor model is usually provided by the system itself and the data of the student model is acquired in execution time. Once the teacher finishes the elaboration of the course, he has to publish the ASWE on a web server.

2. **Use of the ASWE.** In this stage students log in and use the system through a web browser; meanwhile in a transparent way the usage information is picked up and stored in students' web server log files.

3. **Knowledge Discovery.** This is the stage of mining for prediction rules. After the log files have been preprocessed and transferred to a database the teacher can apply knowledge discovery algorithms (classical or evolutionary algorithms) to obtain important relationships among the picked up data. In our case, we want to discover relations between students' knowledge levels, times and scores. To do this task, a generic data mining tool like Weka (Witten and Frank, 2000) can be used, or a specific visual tool like EPRules (Romero et al., 2002).

4. **Improving the ASWE.** Finally the teacher, using the discovered relationships, carries out the modifications he considers to be most appropriate to improve the ASWE. For instance he can modify the course's structure (joining concepts, changing concepts from level or chapter, etc.) and content (eliminating or improving bad questions,

bad content pages, etc.). To do this, he can again use an authoring tool.

The previously described process can be repeated as many times as we want, but it must be done when we have enough new students' usage information. So, it is an iterative process and the ASWE can be progressively improved as more students use it.

The whole process of knowledge discovery with ASWEs can be carried out by the teacher or author of the course, using the EPRules tool (Romero et al., 2002). Figure 2 presents the specific process, beginning with the selection and pre-processing of course usage data, and proceeding with visualizing the rules discovered by applying data mining algorithms. Rules selected by the author can then be used to improve the course.

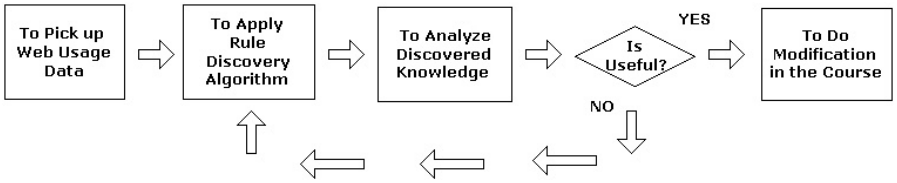


*Figure 2.* Specific knowledge discovery process

The discovery process always begins with the selection of the database where the pre-processed usage data of the course to be used are stored. (These data are taken from log files, preprocessed and then stored in a database.) Then the knowledge discovery algorithms to be applied must be selected as well as their specific parameters and both the objective and subjective restrictions we want the discovered rules to fulfill. After finishing the algorithm execution, the group of discovered prediction rules is displayed: the elements of the rule antecedent and consequent as well as the evaluation measures of each rule are shown, and it is determined if the group of discovered rules is interesting or not. This depends on the number of rules, on their quality with respect to the different measures, and on their semantic meaning. Then it is decided which of the discovered rules are interesting enough to be used to take decisions on possible modifications in the course. If the rules are not considered sufficiently interesting the algorithm is applied again, with different parameters and restrictions, in order to discover a more interesting group of rules. The whole process is carried out in a direct way from the EPRules tool (Romero et al., 2002), a graphic tool developed specifically to solve the problem of prediction rule discovery in ASWEs.

In order to show the success of this methodological proposal we have to prove that there are algorithms for knowledge discovery that

can obtain useful information for the improvement of the system. But we also need an ASWE for doing the tests with a real system. We have developed a Linux course using AHA! (De Bra et. al., 2003) which is the "Adaptive Hypermedia Architecture". We have made some modifications to AHA! in order to increase its adaptation power in education (Romero et al., 2002). More precisely, we want to adapt or personalize the system to each particular student depending on his/her knowledge level.

## 4. Prediction Rule Discovery

The IF-THEN rule is one of the most popular forms of knowledge representation, due to its simplicity, compressibility and expressive power (Klösgen and Zytkow, 2002). Depending on the knowledge it stores, there are different types of rules. In this way, they are referred to as: decision rules, association rules, classification rules, prediction rules, causal rules, optimization rules, etc. In the area of knowledge discovery in databases, the most studied ones are association rules, classification rules and prediction rules. One example of the generic format of IF-THEN rules in EBNF (Extended Backus Naur Form) is shown in Table I.

Table I. Example of the IF-THEN rule format

| | | |
|---|---|---|
| <rule> | ::= | IF <antecedent> THEN <consequent> |
| <antecedent> | ::= | <condition> + |
| <consequent> | ::= | <condition> + |
| <condition> | ::= | <attribute> <operator> <value> |
| <attribute> | ::= | Each of the possible attributes of the set |
| <value> | ::= | Each of the possible values of each attribute domain |
| <operator> | ::= | = \| > \| < \| ≥ \| ≤ \| ≠ |

### 4.1. ASSOCIATION RULES

The objective of association rules (Agrawal et al., 1993) is to look for relationships among attributes types in databases, taking place in the antecedent and consequent of the rules. Association rules are typically used in e-commerce to model the clients' preferences and purchases. These rules have the format: IF "user acquires the product A" THEN "user also acquires the product B" with values of *support* and *confidence* (Agrawal et al., 1993) greater than a user-specified minimum

threshold. In the more general form of these rules, the rule antecedent and consequent can present more than one condition. The confidence of the rule is the percentage of transactions that contain the consequent among transactions that contain the antecedent. The support of the rule is the percentage of transactions that contain both antecedent and consequent among all transactions in the data set.

### 4.2. CLASSIFICATION RULES

The objective of classification rules (Quilan, 1987) is to obtain knowledge in order to create a classification system (similar to a classification tree). In the antecedent of the rule there are some requirements (in form of conditions) that should match a certain object so that it can be considered to belong to the class that identifies the consequent of the rule. From a syntactic point of view, the main difference with association rules is that they have a single condition in the consequent which is the class identifier name.

### 4.3. PREDICTION RULES

The objective of prediction rules (Noda et al., 1999) is to predict an objective attribute depending on the values of another group of attributes. Its syntax is similar to classification rules that have only one condition in the consequent, but now it is similar to any other condition (about the value of the attribute to predict). And any of the attributes would appear in the consequent of the rule as it occurs in association rules.

We are going to discover prediction rules through a dependence modeling task. This data mining task consists of the prediction of relations between attributes specified or not by the user (Freitas, 2002). Prediction rules are very popular in data mining because they usually represent discovered knowledge at a high level of abstraction and it can be used directly in the decision making process. Dependence modeling can be considered as a generalization of discovering classification rules or a specialization of discovering association rules. However, the task of dependence modeling involves a wider notion of dependence than classification does, and it is usually associated with a much wider search space. Also, the classification task is very asymmetric with regard to the attributes, since the objective attribute or class can only occur in the consequent and the prediction attributes in the antecedent. And although the association task is symmetric with the attributes, like with prediction rules, several attributes may occur at the same time in the rule consequent. Another difference is that association rule discovery tries to find only the rules with at least some minimal support and confidence.

The typical knowledge discovery process we are going to perform is shown in Figure 3.


*Figure 3.* Generic knowledge Discovery Process

The typical knowledge discovery process we are going to perform is shown in Figure 3. As we can see, data mining is only one step of the full process of knowledge discovery (Mitra et al., 2001) that consists of:

**Preprocessing.** This consists of the data gathering, data cleaning, conversion of continuous data, attribute selection, data integration, etc. We have to preprocess students' usage information picked up during their use of the course and stored in log files. We have to identify the specific information for each student, too.

**Data Mining.** This consists of the application of a data mining task: classification, regression, clustering, rule discovery, etc. We are going to do prediction rule discovery.

**Postprocessing.** This consists of the interpretation, evaluation of the obtained results and the utilization of the discovered knowledge. We are going to use this information with the objective of making decisions in order to improve ASWEs.

In the following sections, we are going to describe each of these stages in detail for our specific problem of improving ASWEs. We then describe our proposed solution of using knowledge discovery with evolutionary algorithms.

## 5. Description of Students' Usage Information

Normally the information used in web mining is the information stored in typical web server log files. But this rough information deals only with visited pages and times of request from a determined internet address. In the case of ASWEs we need a higher level of information about each student (Heift and Nicholson, 2000). So, we have to enhance server log files adding information about: scores obtained in activities, times of reading pages, knowledge level obtained in concepts and chapters, etc.

We have developed an adaptive web-course of LINUX using AHA! version 1.0 (Romero et al., 2002) to obtain the usage information we

need. AHA! is a general architecture for developing adaptive hypermedia applications. We have chosen AHA! to build up and improve our course because: (a) it lets us convert any type of web-based applications into adaptive versions, (b) it stores usage information in log files, (c) it has an adaptive engine which uses adaptation techniques (conditional inclusion of fragments and hiding or annotation of links) and (d) its source code is available (De Bra et. al., 2000). It has been necessary to modify AHA! in order to be able to perform the adaptation depending on the knowledge level of each particular student. To do this, we have performed the following changes (Romero et al., 2002) in:

**Domain Model** . We use the typical structure of educative material by adding levels. So, a course consists of several chapters organized in lessons that have several concepts divided in difficulty levels (we use three levels: HIGH, MEDIUM and LOW).

**User Model** . We have implemented students' knowledge for each concept and each chapter through discrete values: 0 (NOT YET READ), 10 (BEGINNER), 50 (NORMAL), 100 (EXPERT).

**Adaptation Engine** . We have performed the adaptation from a chapter view point (Figure 4) and it consists of: Before starting a new chapter students have to do an initial adaptive test to obtain their initial knowledge level. Then the system presents only the concepts which have the appropriate difficulty level. Next, the students have to read the exposition content pages and perform the evaluation activities for each concept. Eventually they have to do a final test to evaluate their knowledge about this chapter. Depending on the obtained level, students may repeat the chapter at the same level (if they obtain a BEGINNER level) or they can go to a higher level of the same chapter or they can pass to another chapter (if they obtain a NORMAL or EXPERT level). For each new chapter, everything previously mentioned starts again: first initial test, then exposition and activities pages, and next the final test.

The specific usage information we are going to use for discovering prediction rules is collected from a course about the operating system LINUX, developed with the modified AHA! system. This course has been taken by 50 students in computer science engineering at the Cordoba University. We have modified AHA! in order to increase its adaptation power in education (Romero et al., 2002). More precisely, we wanted to adapt or personalize the system for each particular student depending on his knowledge level. In figure 5 the same chapter of the course is shown at three different difficulty levels (with different

concepts and background color): beginner(left), normal(middle) and expert(right).

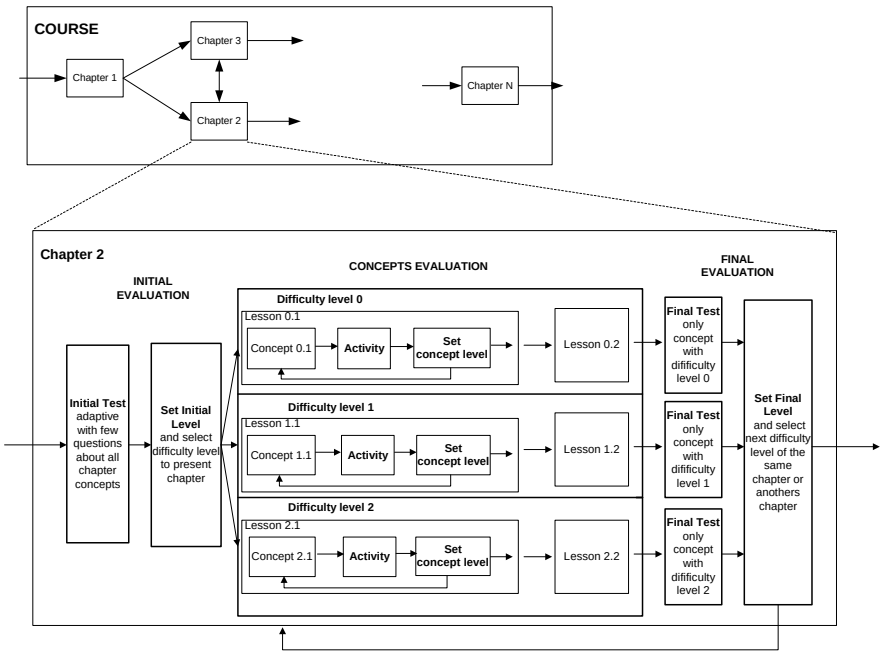


*Figure 4.* Modified AHA! adaptation engine

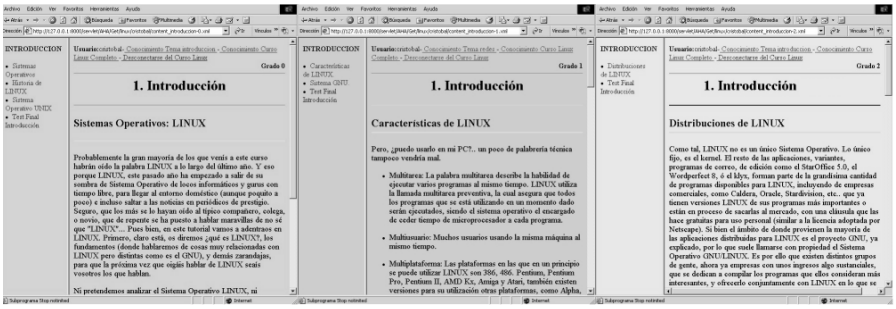


*Figure 5.* Three different levels of the Introduction Chapter of the Linux course.

The original AHA! (1.0) stores usage information for each student in two log files (called "log" and "model") in which information about user navigation and user knowledge are stored. We have added another log file (called "test") to store the scores of activities and test questions. The specific content of these three files for each student are:

**Log** is a text file that contains information about the name and the time (in seconds) of each visited page.

**Model** is an XML file that contains information about the knowledge levels students have for each concepts and chapter, in a numerical form (value 0, 10, 50 or 100).

**Test** is a text file that contains information about the success or failure that students have in the test or activities questions (YES or NO respectively).

### 5.1. Data Preparation

Before we can apply data mining algorithms we have to transform and move all information stored in the previous logs files into a (relational) database. We have done it to facilitate and to increase the speed of algorithms. During this process we have carried out the following preprocessing task (Freitas, 2002): attribute selection, data cleaning, discretization of continuous attributes and data integration.

#### 5.1.1. *Attribute selection*

The main goal of attribute selection is to select a subset of relevant attributes, out of all available attributes of the data being mined. We have selected the attributes in a manual way. The chosen ones are: *user name* (name students), *course* (name of the course), name (of the element), *type* (exposition page, activity page, test page, concept or chapter), *difficulty* (assigned navigation level), repetition (number of repetitions), *time* (interval time used), *score* (obtained success or failure), *level* (obtained knowledge level).

#### 5.1.2. *Data cleaning*

This consists of looking for and rejecting erroneous, duplicate and irrelevant data. We have discovered several types of errors: high times (higher than 10 minutes), incomplete data (incompletely visited chapters and unfinished tests and activities). And we have also discovered several types of irrelevant data: container pages (frame pages), index or content pages, help pages and log out pages. We have rejected all this information.

#### 5.1.3. *Transforming continuous attributes*

This consists of transforming continuous attributes into discrete attributes that can be treated as categorical attributes. The basic idea is to partition the value domain of a continuous attribute into a small number of intervals. We can choose among the following unsupervised global methods (Dougherty et al., 1995): the equal-width method, equal-frequency method or the manual method (in which you have to specify

the cut-off points.). We have only transformed one attribute: time. We have assigned to it three values or labels: HIGH, MEDIUM and LOW, using the equal-width method.

5.1.4. *Data Integration*
The goal of data integration is to group together all the data that come from different sources making up a data warehouse. In our case, we have gathered all preprocessed data from the log files (log, model and test) of each student in a MySQL relational database (Dubois, 2002). We have used MySQL because is portable, fast and freeware. In Figure 6 we show the relational scheme of the students' usage information database we have used.

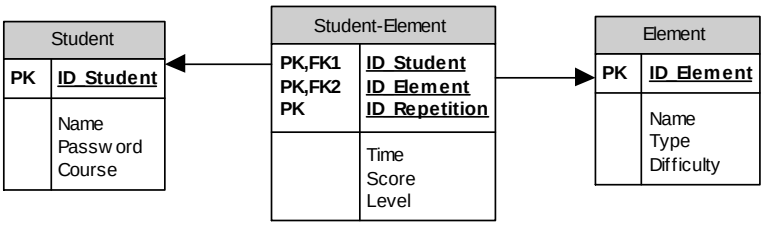


*Figure 6.* Database Relational Scheme

## 6. Evolutionary Algorithms for Rule Discovery

The task of rule discovery has been approached using different paradigms: construction of decision trees, inductive learning, instance-base learning and more recently neural nets and evolutionary algorithms (Witten and Frank, 2000). The construction of decision trees algorithms is the most used at the moment in data mining. They are very fast and surprisingly effective to find precise classifiers. But, as they use greedy heuristics to divide data, they may fail to find some multi-variable relationships. Inside the conventional rule learning algorithms there is a wide variety of alternatives whose common characteristic is to perform a more meticulous search than the previous ones. On the other hand evolutionary algorithms are able to carry out many meticulous searches. They can do an implicit step back in the search of rule space that will allow to find complex interactions among attributes that other types of algorithms are not able to find.

Evolutionary algorithms are a paradigm based on the Darwin evolution process (Darwin, 1859), where each individual codifies a solution and evolves to a better individual by means of genetic operators (mutation and crossover). The main advantages in adopting EAs are (Dhar

et al., 2000) the ability to work in a search space thoroughly, and the ability to allow arbitrary fitness functions in the search. Their main disadvantages are lack of speed and randomness in creating the initial population. The main motivation to use evolutionary algorithms for rule discovery is that they perform a global search and cope better with attribute interaction than greedy rule algorithms commonly used in data mining. Most data mining methods are based on the rule induction paradigm, where the algorithm usually performs a kind of local search. Also, the fitness function in evolutionary algorithms evaluates the individual as a whole, i. e. all the interactions among attributes are take into account. In contrast, most rule induction methods select one attribute at a time and evaluate partially constructed candidate rules, rather than full candidate rules.

We can view rule prediction discovery with evolutionary algorithms from two different viewpoints: restricted or unrestricted. In the restricted way (Freitas, 2002), the problem is treated as classification rule discovery in which users have to specify the attribute or attributes to be predicted. So, individuals only represent the rule antecedent conditions. The objective is to discover the best conditions that predict the previously set attributes. In the unrestricted way (Romero et al., 2002), the problem is treated as association rule discovery. And the individuals represent complete rules with the antecedent and the consequent condition. In this case the objective is to discover the best rules that predict any attribute. We are going to use the unrestricted way, but users can specify some filters to find certain types of rules (with a concrete type of condition in the rule antecedent or consequent, a maximum number of conditions in the rule, etc.)

### 6.1. Grammar Based Genetic Programming for Discovering Prediction Rules

There are different paradigms of Evolutionary Algorithms (EA): Evolutionary Programming (EP), Evolutionary Strategies (ES), Genetic Algorithms (GA) and Genetic Programming (GP). But the most used EA to solve rule discovering problem are GA and GP.

Genetic Algorithms (GA) for rule discovery can be divided into two main approaches (Freitas, 2001), based on how rules are encoded in the population of individuals:

**Michigan approach.** In this approach, each individual encodes a single prediction rule.

**Pittsburgh approach.** In this approach, each individual encodes a set of prediction rules.

The most used approach is Michigan, in which an individual is usually a linear string of rule conditions, where each condition is often an attribute-value pair. This approach makes the individual encoding simpler and syntactically shorter. On the other hand, with the Pittsburgh approach the individual encoding is more complicated and syntactically longer, but the fitness of an individual can be evaluated by considering its rule set as a whole.

Genetic Programming (Koza, 1992) is like a version of Genetic Algorithms that uses trees to represent individuals. In fact, the algorithms of both approximations are the same. And although GA is a most widely used, GP can be considered as a more open-ended search paradigm. In general, GP has a higher expressivity and can discover interesting and surprising rules (Gilbert et al., 1998). The search performed by GP can be very useful, since it can produce many different combinations of attributes. A basic genetic programming system consists of five components (Koza, 1992): representation for programs or genome structure, a procedure to initialize a population of programs, a fitness function to evaluate the performance of the program, genetic operators, and parameters.

In GP an individual is usually represented by a tree, with rule conditions and/or attributes values in the leaf nodes and functions in the internal nodes. But there is a problem when encoding rules into a GP individual, due to the closure property of GP (Freitas, 2001), which requires that the output of a node can be used as the input to any parent node in the tree. There are different approaches of GP that cope with the requirement of closure: Strongly Typed Genetic Programming (STGP) or Constrained-Syntax Genetic Programming, Grammar Based Genetic Programming (GBGP), etc.

The GBGP or Grammar Guided Genetic Programming (GGGP) represents individuals as a derivation tree of a grammar defined by the user to specify the problem solution space (Whigham, 1995). GBGP systems use grammars to set syntactical constrains on programs. The use of grammars also helps to overcome the closure requirement in canonical genetic programming, which cannot always be readily fulfilled. The grammar can be used to enforce elaborate semantic restrictions based on the domain knowledge provided by a domain expert.

We have chosen GBGP due to its expressivity and capacity to interact with the user in which he can select different types of desired rules by restricting only the grammar. GBGP has demonstrated high performance on a number of problems and it has been considered one of the most promising areas in the field of research on genetic pro-

gramming (Nordin et al., 1998). Next, we are going to describe several examples of using Genetic Programming for rule discovery.

One of the first works that use GP to perform knowledge discovery is the MASSON system (Yyu and Eick, 1996). It is centered on the problem of discovering the common characteristics that are shared by a set of objects belonging to an object-oriented database. The commonalities between a set of objects are specified using object-oriented queries. MASSON employs GP to search interesting queries and evaluate them to see whether queries compute the same set of objects given by the user.

Another related work is done by Ratle and Sebag (Ratle and Sebag, 2000) who uses genetic programming for machine discovery of empirical laws. They propose a way of enforcing dimensional constraints through formal grammars, to restrict the GP search space to dimensionally admissible laws. They use grammar rules for incorporating dimensionality constraints in GP and they use an initialization procedure based on a dynamic pruning of the grammar, in order to generate only feasible trees of prescribed derivation depth.

A different work, that uses genetic programming to discover classification rules for diagnosing certain pathologies in medical databases is done by Bojarczuk et. al (Bojarczuk et al., 2001). They use constrained-syntax GP, in which some restrictions should be considered in order to have a valid rule. They use databases of medical domains: chest pain, dermatology and breast cancer, for discovering high level, comprehensible classification rules.

One specific example of data mining using grammar based genetic programming is done by Wong and Man (Wong and Leung, 2000) in LOGENPRO (The LOGic grammar-based GENetic PROgramming system). They use Inductive Logic Programming (ILP) and genetic programming to set syntactical and semantic restrictions . They describe a framework, called GCP (Generic Genetic Programming), that integrates genetic programming with a formalism of logic grammars. This formalism is powerful to represent sensitive information and domain-dependent knowledge. This knowledge can be used to increase the learning speed and/or improve the quality of the knowledge induced.

Finally, other work that use a genetic programming framework for two data mining tasks: classification and generalized rule induction is done by Freitas (Freitas, 1997). He emphasizes the integration between a GP algorithm and relational database systems. This integration leads to minimization of data redundancy and to improvement of scalability, data-privacy control and portability.

As we can see, most of the works using Genetic Programming for rule discovery are focused on classification rules. In this specific approach

the rule consequent is a single name (the name of the class) and not a typical condition (attribute-value pair). Due to this, many approaches only encode the antecedent of the rule in the individuals. But our approach is more general, in the sense that we want to discover prediction rules, more generally than classification rules (See Section Prediction Rule Discovery). The main difference in our approach is that we encode together in the individuals both the antecedent and the consequent of the rules.

Next, we are going to describe our approach to Grammar Based Genetic Programming for discovering prediction rules.

### 6.2. Individuals Encoding

In individuals encoding with Evolutionary Algorithms we have to distinguish between the phenotype and the genotype of individuals, especially in our GBGP approach:

**Genotype.** The genotype is the genetic composition of the individual. The genotype of our individuals is a syntax tree of instructions, which we have implemented as an integer list (Ventura et al., 2002).

**Phenotype.** The phenotype is the meaning of the genetic material for the user. The phenotype of our individuals are prediction rules. The meaning of these rules is provided by a grammar. Each individual generated by the GBGP is evaluated against the database using several queries in order to compute the contingency table (see Appendix A. Rule Evaluation Measures).

The grammar we have used to generate the individuals that represent prediction rules is shown in Table II. Prediction rules consist of an antecedent with several conditions and consequents with only one condition. Each condition relates to an attribute (about time, success and level) with one possible value of this attribute. We have not shown all the terminal symbols of valid attribute names because there are a lot of them (all the names of web-pages, questions, activities and tests).

All the values of the attributes are categorical or nominal. In Table III we show the functions or non-terminal symbols we have used in our grammar.

Table II. Rule Prediction Grammar in EBNF

| | | |
|---|---|---|
| <rule> | ::= | IF <antecedent> THEN <consequent> |
| <antecedent> | ::= | <antecedent> AND <condition> \| <condition> |
| <consequent> | ::= | <condition> |
| <condition> | ::= | <level-attribute> = <level-value> \| |
| | | <time-attribute> = <time-value> \| |
| | | <success-attribute> = <success-value> |
| <level-attribute> | ::= | LEVEL.Name of a valid level attribute |
| <time-attribute> | ::= | TIME.Name of a valid time attribute |
| <success-attribute> | ::= | SUCCESS.Name of a valid success attribute |
| <level-value> | ::= | BEGINNER \| NORMAL \| EXPERT |
| <time-value> | ::= | HIGH \| MEDIUM \| LOW |
| <success-value> | ::= | YES \| NO |

Table III. Functions and arguments

| **Functions** | **Input arguments** | **Output arguments** |
|---|---|---|
| IF THEN | boolean | boolean |
| AND | categorical | boolean |
| = | categorical | categorical |

In Figure 7 an example rule is shown and the derivation tree generated by our grammar to represent this rule.

The rule in Figure 7 shows that students, evaluated as EXPERT in the concept CHARACTERISTIC in the INTRODUCTION chapter at the HIGH level, fail question number two of the activity of this concept, and they also need a HIGH time to answer that question. So, this rule shows that something is wrong with this question and the teacher should review it (in relation to the information that is supposed to prepare the students for the question).

### 6.3. Evolutionary Algorithm

The evolutionary algorithm we have implemented to discover prediction rules is a generational GP with a external elitism file. So, we use two different groups of individuals. The first one represents the current population in each generation, and the second one represents the elite population or best individuals that finally will be returned to the user

**IF** SUCCESS.characteristic-introduction-high(2)=NO **AND**
TIME.characteristic-introduction-high(2)=HIGH
**THEN**
LEVEL.characteristic-introduction-high=EXPERT

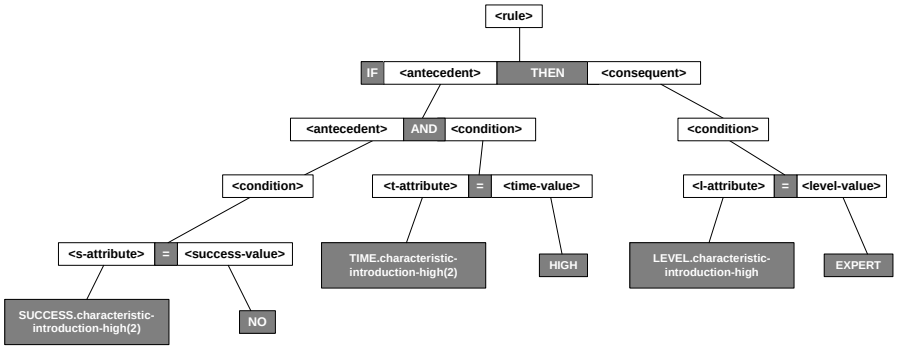


*Figure 7.* Derivation tree of example rule

as a set of discovered rules. In Table IV we show the evolutionary algorithm.

Table IV. Evolutionary Algorithm

**Begin**
Generate an initial population P from selected data
Create an empty file E
**While** (current-generation < max-generation) **do**
Select parents from P and E individuals.
Apply genetic operator over selected parents.
Evaluate obtained children with multi-objective or single-objective approach.
Update individuals in P and add the best news individuals to E.
Increase current-generation by one.
**End While**
**End**

The first step in the algorithm is to create the initial population P (with fixed size) from the initial data selected by the user (see Section Initialization). We have also to create an external file E (with variable size) that is empty initially, but we will store in E the best individuals of the current population in each generation step. Then we select the parents from the current population P and elite file E to reproduce (see Section Selection), although the first time we select them only

from P. The children are generated by applying genetic operators (see Section Genetic Operators) and later they are evaluated using a multi-objective or single-objective approach (see Section Evaluation). Next the elite population is updated adding the best new individuals of the current population P. The process finishes when a maximum number of generations are done.

Next we are going to describe separately each component of our GBGP algorithm.

### 6.4. Initialization

Initialization consists of generating a group of initial rules. First, the teacher has to select which data he wants to use to compose the initial rules. He can choose to use: all available values, a range of values (those with a relative frequency greater than a specific threshold), frequent or infrequent values (see Figure 8). These initial elements are used only to compose initial rules.


*Figure 8.* Group of initial elements

There are two reasons for allowing the use of different initialization data. The first is to compare the algorithm's performance with different types and number of data: (a) a large amount of data (namely all data), (b) a small amount of data that is the most frequent data and (c) an average number of data that is the range data. And the second is that it can be more logical not to use all data, but to use a representative data set. We propose to use range data, better than only very high frequent (that almost all students match) or very low frequent (that almost any student match) data.

After this, we compose the rules from these initial data, choosing randomly what elements or data are going to be set in the antecedent and consequent of the rules.

The size of the rule varies dynamically depending on the number of elements in the antecedent. The last element always represents the consequent. The user can specify a maximum size for all the rules. After creating the initial population and the other new populations (applying a genetic operator) in each generation, we have to verify that the rules are always correct. Although all rules generated by our grammar (see Table II) are syntactically correct, it is possible that some of them may be semantically incorrect. For example, rules with the same condition

in the antecedent and consequent of the rule, or rules with a repeated condition in the antecedent of the rule. This problem is due to the fact that we use a free-context grammar. To solve it, we repair incorrect individuals by mutating the causing conditions (again) until the rules are correct, before individuals are evaluated.

The elite population is generated in a different way depending on the evolutionary algorithm used: mono-objective or multi-objective. In the case of using only one evaluation measure, we set a threshold so that the individuals with a higher value will be added to the elite group. And in the case of using several evaluation measures at the same time, we use approaches based on the concept of Pareto Front (Fonseca and Fleming, 1993), in which non-dominated individuals are always chosen to be added.

### 6.5. Genetic Operators

The genetic operators we have used are selective crossing, selective mutation (Whigham, 1995) and reparation.

#### 6.5.1. *Selective crossing*

Selective crossing is analogous to crossing trees in genetic programming, in which two subtrees of each parent tree are mixed to form two new child trees. But in selective crossing, the crossing point has to be selected from non terminal symbols and has to be the same in both subtrees to be crossed. Also, the subtrees to exchange have to be semantically compatible. In our specific rules the selective crossing can carry out five different operations: exchange rule antecedents, exchange rule consequents, exchange rule conditions, exchange rule condition attributes and exchange rule condition values. We can vary the probability of each operation when we configure the parameters of the algorithm.

#### 6.5.2. *Selective mutation*

Selective mutation is also analogous to mutation trees in genetic programming, in which a subtree of a tree is mutated to create a new tree. But selective mutation rebuilds only a specific subtree that has a non-terminal root node. This operator maintains the population diversity and we can also vary the probability of each non terminal symbol to be root node in the mutation.

#### 6.5.3. *Reparation*

Reparation is a specific operator to repair incorrect individuals. An individual can be syntactically correct (generated by the grammar) but

semantically incorrect to the user (one rule with the same condition in the antecedent and consequent, or with duplicate conditions). To solve this problem we use the reparation operator which change the conflicting condition by applying a simple mutation of the condition until the problem disappears.

### 6.6. Evaluation

Evaluation consists of calculating the fitness function, i. e. the quality evaluation function of the current rules. There are a lot of evaluation rule measures (Tan et al., 2002) (Lavrac et al., 1999) (Yao and Zhong, 1999) that come from statistics, machine learning and data mining (see Appendix A: Rule Evaluation Measures), each of them trying to evaluate one feature of the rule (precision, interest, reliability, comprehension, simplicity, etc.). But there is no single measure that is the best in all the application domains. The previous considerations suggest to use several measures, similar to a problem of multi-objective optimization (Fonseca and Fleming, 1993). In this case, there isn't a single aptitude function, but rather there are several functions to optimize simultaneously. There are different approaches to solve the problem of the multi-objective optimization with evolutionary algorithms: one approach is to use aggregation functions, another is to use the concept of Pareto Front.

#### 6.6.1. *Aggregation Function*

In this case, the evaluation function is a linear combination of different measures to optimize (Deb, 2001). The weight of each component in the linear combination represents the relative importance of each single measure in the global function. There are several examples of aggregation functions used in the task of rule discovery; some of them are:

**Aggregation function proposed by Araujo** (Araujo et al., 1999) consists of two components (Equation 1): the first one uses the J-measure (Smythe and Goodman, 1992) that is related to the interest of the rule, and the second uses the number of potential attributes of the antecedent.

$$Fitness(A \rightarrow C) = \frac{w1 * J1 * w2 * \frac{n_{pu}}{n_T}}{w1 + w2} \tag{1}$$

where J1 is the one-sided variant of the J-measure, $n_{pu}$ is the number of potentially useful attributes in the antecedent, $n_T$ is the total number of attributes in the antecedent, and w1, w2 are user-defined weights.

**Aggregation function proposed by Liu** (Liu and Kwok, 2000) consists of three components (Equation 2): the first one is the Laplace (Bayardo and Agrawal, 1999) to measure rule consistency, the second represents the rule completeness and the third one represents the rule generality (Liu and Kwok, 2000).

$$Fitness(A \rightarrow C) = w1 * Lap(A \rightarrow C) + w2 * \frac{p(AC)}{p(C)} + w3 * Simp(A \rightarrow C) \quad (2)$$

where Lap is the Laplace measure, p is the relative frequency, Simp is the simplicity measure which decrements when the number of conditions in the rule antecedent increases, and w1, w2 and w3 are user-defined weights.

**Aggregation function proposed by Freitas** (Freitas, 2002) consists of two components (Equation 3): the first represents the rule accuracy and the second the rule comprehensibility (Liu and Kwok, 2000).

$$Fitness(A \rightarrow C) = w1 * \frac{p(AC)}{p(AC) + p(A\neg C)} + w2 * Simp(A \rightarrow C) \quad (3)$$

where p is the relative frequency, Simp is the simplicity, and w1 and w2 are user-defined weights.

6.6.2. *Pareto Front*

The algorithms based on the concept of Pareto Front (Fonseca and Fleming, 1993) use a vector of objectives to optimize within each individual. The purpose is to make population converge towards the group of best solutions denominated as Pareto Front. The solution is the best in terms of all objectives together and not in any specific objective. There are different types of algorithms inside this approach, some of them are:

**MOGA.** The algorithm MOGA (Multi-Objective Genetic Algorithm) (Fonseca and Fleming, 1993) is based on the idea of ordering individuals depending on their non-dominance. The order (rank) of each individual corresponds to the number of individuals by which it is dominated. The non-dominated individuals have an order value of one, while the rest are penalized according to the number of individuals by which they are dominated. An individual is dominated by another individual if it is equal or worse in some of the objectives.

**NSGA.** The algorithm NSGA (Non-dominated Sorting Genetic Algorithm) (Srinivas and Deb, 1994) is based on several steps of classification of individuals. It also establishes ranges between individuals based on their non-dominance. First the population is ordered using the non-dominance concept. Then the aptitude is assigned to each individual depending on its range inside the population and using an aptitude sharing method.

Finally, in all evolutionary algorithms based on the concept of Pareto Front, it is necessary to choose the specific objectives to use. In our case, we have to choose what the rule quality evaluation measures are which we want to optimize. According to some research (Freitas, 2002) the discovered knowledge by a data mining algorithm should satisfy three main aspects: it should be accurate (certainty), interesting (novel, surprising, useful) and comprehensible (simple). We have used three criteria to measure the quality of the rules:

**Accurate** . The concept of rule accuracy (Lavrac et al., 1999) we have used is the same as confidence in association rule mining, in which rule accuracy measures the reliability of the rule in the prediction of positives cases, since it measures the correctness of returned results. So we measured the accuracy of the discovered rules using the whole data set, as is done in association rule mining and not using different test and training sets as done in classification.

**Interesting** . Rule interestingness (Piatesky-Shapiro and Metheus, 1994) can be measured using two types of measures: subjective (user-driven) (Silberschatz and Tuzhilin, 1995) or objective (data-driven) (Tan et al., 2002). We have used user constraints about the knowledge he wants to discover and an objective rule interestingness measure.

**Comprehensible** . The discovered knowledge must be comprehensible (Askira-Gelman, 1998) to the user. To achieve this goal we have used a high level knowledge representation (using IF-THEN rules), we measure the size of the rule (number of conditions) and we count the number of discovered rules.

For this reason, we have used a vector of three values where each one measures one of these aspects. The specific measures we have used are:

- **Certainty Factor**. The Certainty Factor (C. F.) (Shortliffe and Buchanan, 1975) is a measure about the rule precision and it can be used instead of the confidence with better results (Delgado et

al., 2001). The certainty factor of a rule $A \rightarrow C$, where A is the antecedent of the rule, and C is the consequent of the rule, is

$$CF(A \rightarrow C) = max(\frac{p(C/A) - p(C)}{1 - p(C)}, \frac{p(A/C) - p(A)}{1 - p(A)}) \quad (4)$$

where max is the maximum function and p is the relative frequency.

- **Interestingness**. The interestingness (Tan et al., 2002) is a measure related to the rule interest that can be better than the classic Piatetsky-Shapiro measure of interest (Silverstein et al., 1998). The interestingness of a rule $A \rightarrow C$, is

$$IS(A \rightarrow C) = \sqrt{I(A \rightarrow C) * \frac{p(CA)}{N}} \quad (5)$$

where I is the Piatetsky-Shapiro measure of rule interest, p is the relative frequency and N is the total number of data instances.

- **Simplicity**. The simplicity (Liu and Kwok, 2000) is a measure about rule compressibility so that the shorter the rule is the more comprehensible. The simplicity of a rule $A \rightarrow C$, is

$$Simp(A \rightarrow C) = (1 - \frac{AntecedentSize}{MaximumSize}) \quad (6)$$

where AntecedentSize is the number of conditions in the antecedent and MaximumSize is the maximum number of condition in the antecedent.

We have selected these tree measures because several referenced works (Delgado et al., 2001), (Tan et al., 2002), (Liu and Kwok, 2000) have proven that they offer insight individually. And our objective is to prove that using them together in a multi-objective function, they can offer a better insight.

### 6.7. Selection

Selection consists of the choice of the rules from the population P and from the elite population E to be parents in the reproduction (by crossing or mutation). We use rank-based selection or linear ranking (Michalski, 1998) that first ranks the population according to its evaluation value (fitness function value) and then every rule receives its final fit from its ranking. Hence, the worst rule (rule with less fitness value) will set fitness 1, the second worst 2, etc. and the best will set fitness N (where N is the number of rules in the population). Parents

are selected according to their fitness. With this method all the rules have a chance to be selected and the probability to select an individual is proportional to its position.

In order to assure diversity in the population we also use a metric for the number of different conditions there are in the antecedent and consequent of each rule. The individuals with a higher value in this metric will be structurally the most different and they will be more probably elected. In the new population we also assure that there are no repeated individuals, in order to avoid the problem of premature convergence.

## 7. Experimental Results

In this section we describe the developed implementation of different knowledge discovery algorithms and the specific software for applying them in the problem area of ASWEs improvement. We will also describe the tests that were carried out and compare the results obtained through the execution of different algorithms.

### 7.1. Implementation

In order to facilitate the realization of the full process of discovery of prediction rules we have developed a specific graphical tool named EPRules (Education Prediction Rules) developed in Java and intended to be used directly by the teacher or the course developer. So, this tool (Figure 9) is much easier to use (for a non-expert data-mining person) than other generic tools such as DBMiner (Klösgen and Zytkow, 2002) and Weka (Witten and Frank, 2000). And as it is a special-purpose tool, it uses specific domain knowledge, and that's why it is more powerful for discovering knowledge than other (general-purpose) tools.

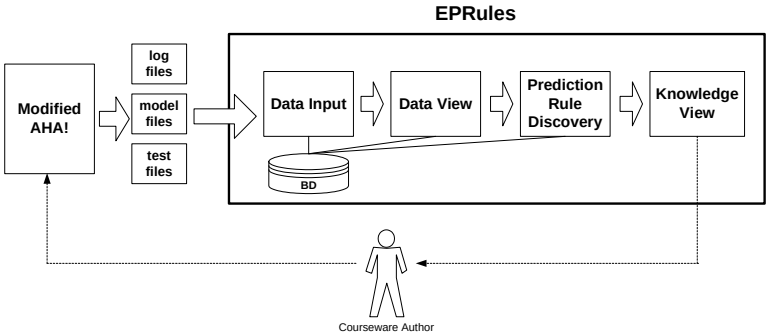


*Figure 9.* EPRules tool

The main components of the EPRule tool interface (Figure 10) are:

**Data input.** It allows either to open a database with course usage data or to create a new one and to add new students usage information and preprocess it. To create a new database or to add data, the course usage files must be selected (students' log files) that will be preprocessed and integrated into a relational database. The algorithm parameters to transform the time-variable can also be selected and configured. (This consists of transforming continuous attributes into discrete attributes). We need to transform the attribute time, assigning it three values: HIGH, MEDIUM and LOW. (Figure 10).

**Data view.** It allows to visualize students' usage data and to carry out some basic statistics (maximum, minimum, average, etc.). These data are about the access times, correct answers and knowledge levels obtained by students for the different web pages (activities and contents) that make up the course. One can select either to visualize all students' data or a specific student's data, or just about a particular theme of the course or about a specific concept of the theme, or the visibility and difficulty level of a particular theme (high, normal, low), or a type of particular information (time, level or correct answers).

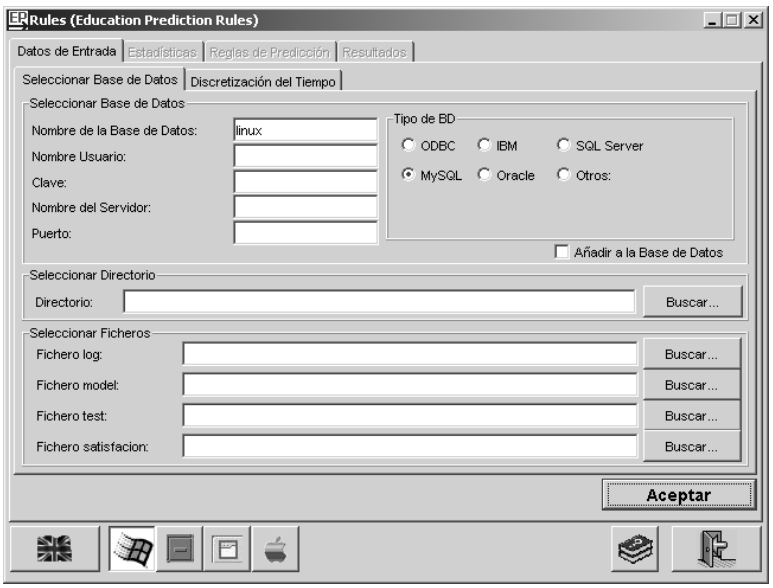


*Figure 10.* EPRules tool

**Prediction rule discovery.** This is the most important part of the tool because this is where the different algorithms for rule discovery are applied. It allows: (a) to select the rule discovery algorithms: ID3 (decision trees constructing algorithm) (Quilan, 1987), Apriori (association rule mining algorithm) (Agrawal et al., 1993), Prism (covering algorithms) (Cendrowska, 1987) and GBGP; (b) to choose the specific execution parameters of each algorithm, (c) to select the subjective restrictions that rules should match (Figure 11), i. e. just one chapter or concept, a single student, a determined knowledge level, score or time, and (d) to choose the objective evaluation function, so that the discovered rules are really useful to the teacher.

**Knowledge view.** It allows to visualize the discovered prediction rules, both antecedent and consequent conditions of the rules and the values for each evaluation measure the rules have (confidence, support, interest, gini, laplace, etc., see Appendix A). It appears automatically after finishing the algorithm execution. In a predetermined way, the rules appear ordered from the first discovered one to the last one, but they can be rearranged taking into account a condition or the value of any measure by simply clicking the desired column.

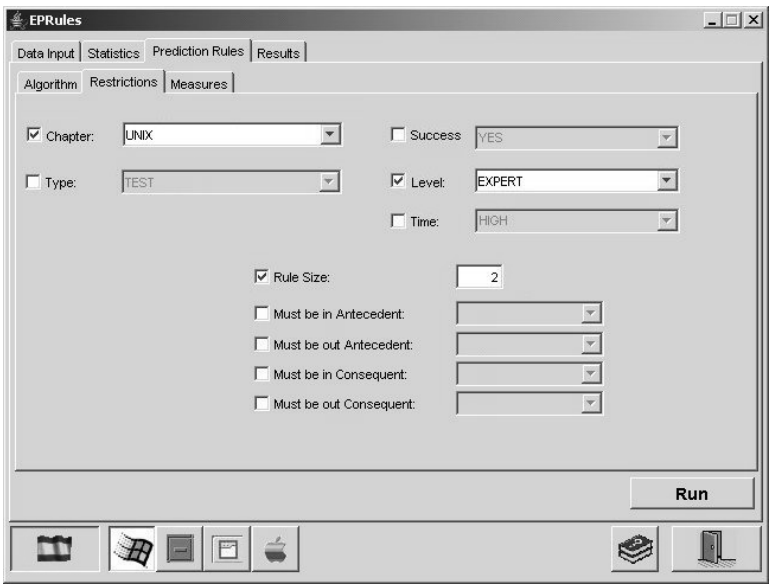


*Figure 11.* Restrictions windows in EPRules tool

The classic algorithms for knowledge discovery (ID3, Prism and Apriori) have been implemented in Java. We have chosen these algorithms since they are the most representative methods in data mining (Witten and Frank, 2000) and previously other authors have proposed and used them in comparison tests with a new proposed algorithm (Freitas, 2002), (Hipp et al., 2000).

We had to adapt the algorithms to the specific discovery of prediction rules. To do this, we had to modify the association and classification algorithms. The conversion from association rules algorithm to prediction rules algorithm is trivial, since it is only necessary to force rule consequents to have a single condition. In the case of converting classification rules algorithms into prediction rules algorithms the attribute of rule consequent or class can't be categorical, but rather it can be any normal condition (an attribute value pair). And also, if we want to extract rules with N different attributes, we have to execute the classification rules algorithm N times, using in each case a different attribute as consequent condition class.

We have only modified these algorithms to obtain rules with the same format of the prediction rules generated by our GBGP algorithm. Our objective was to test the quality (using three objectives measures) of the rules obtained by a GBGP in relation to classic algorithms without modifications.

The GBGP algorithms (aggregation functions or based on Pareto Front) have also been implemented in Java using Jclec, a Java Class Library for Evolutionary Computation (Ventura et al., 2002). The GBGP implementation in the Jclec library codes syntactic trees as vectors of ordered integers that represent the tree in a pre-order way. In order to evaluate the individuals it is necessary to transform the list of integers in several SQL (Structured Query Language) queries (Sarawagi et al., 1998) in order to determine the values of the evaluation measures used in each case.

### 7.2. Description of the experiments

We have used the usage data of 50 users of a LINUX course. We have carried out two types of tests to compare the results of each of the implemented knowledge discovery algorithms. The first one is used to compare the number of discovered rules and the second one is to compare the quality of these rules.

We have also performed three more tests on populations of different size. More precisely we have used (Figure 8): all available data, only frequent data (those data with a relative frequency greater than 0.5)

and only range data (those data with a relative frequency greater than 0.2 and lower than 0.9).

In the case of evolutionary algorithms, we have repeated the execution 10 times, using the parameters shown in Table V.

Table V. Evolutionary Algorithm Parameters

| | |
|---|---|
| **INITIALIZATION STAGE** | |
| **Size of Population** | 50, 100 and 200 individuals |
| **Initialization method** | Ramp based initialization |
| Minimum number of productions | 9 |
| Maximum number of productions | 18 |
| **REPRODUCTION STAGE** | |
| **Selection method** | Rank based selection |
| **Crossover Operator** | Selective crossover |
| Probability of success | 0.8 |
| $< antecedent >$ roulette value | 1 |
| $< consequent >$ roulette value | 1 |
| $< condition >$ roulette value | 4 |
| $< attribute >$ roulette value | 2 |
| $< value >$ roulette value | 2 |
| **Mutation Operator** | Selective mutation |
| Probability of success | 0.2 |
| $< antecedent >$ roulette value | 1 |
| $< consequent >$ roulette value | 1 |
| $< condition >$ roulette value | 1 |
| $< attribute >$ roulette value | 1 |
| $< value >$ roulette value | 1 |
| **STOP STAGE** | |
| **Maximum number of generations** | 50 |

— In order to compare classic algorithms versus evolutionary algorithms in similar conditions (without advantage in any of them), we have used for all algorithms: the same form of rule (prediction rule), the same initial data (group of initial elements), the same initial restrictions (size of rule, etc.), and we have used typical values in parameters for each algorithms which need them (minimal support and confidence, number of evolutions, etc.) —

### 7.3. Comparison of algorithms' performance

Using EPRules we have compared the number of discovered rules and the percentage of accurate, interesting and comprehensible rules, obtained through the different implemented algorithms. We have done two comparisons: classic algorithms (ID3, Prism and Apriori) versus an average value of evolutionary algorithms (EA-GBGP), and the previous evolutionary algorithms among them (Liu, Freitas, Araujo, MOGA, NSGA).

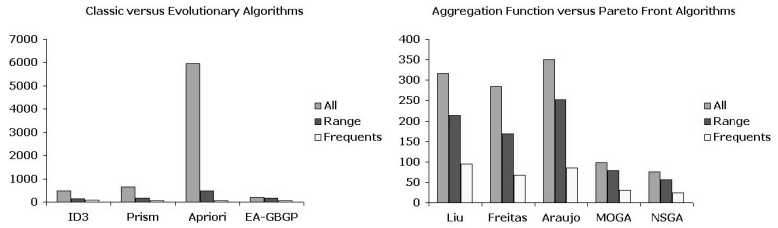


*Figure 12.* Number of Discovered Rules

**Number of Discovered Rules.** Figure 12 shows the total number of rules discovered with classic versus evolutionary algorithms and different versions of GBGP. As we can see in Figure 12 the huge number of rules discovered by classic algorithms isn't useful in the case of using all data, especially with the Apriori algorithm. This effect is attenuated by decreasing the size of used data (range and frequents data). On the right-hand side, we see that the algorithm that discovers the lowest number of rules is NSGA, followed by MOGA. (Note the scale difference between the left and right diagram.)

**Percentage of Accurate Rules.** Figure 13 shows the percentage of accurate rules (rules with a value of certainty factor greater than 0.7) discovered with classic versus evolutionary algorithms and different versions of GBGP.

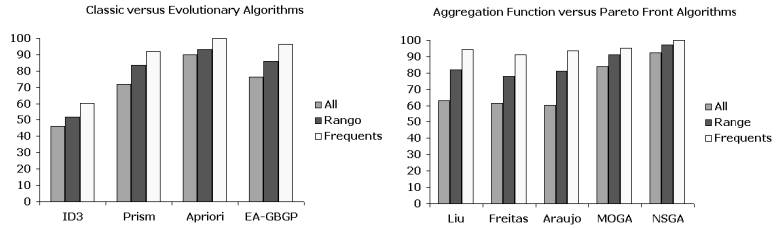


*Figure 13.* Percentage of Accurate Rules

Consulting the information of Figure 13 we can see that the Apriori algorithm discovers very accurate rules. It is also important to point out that algorithms based on Pareto Front (MOGA and NSGA) discover very accurate rules too.

**Percentage of Interesting Rules.** Figure 14 shows the percentage of interesting rules (rules with a value of interestingness greater than 0.5) discovered with classic versus evolutionary algorithms and different versions of GBGP.

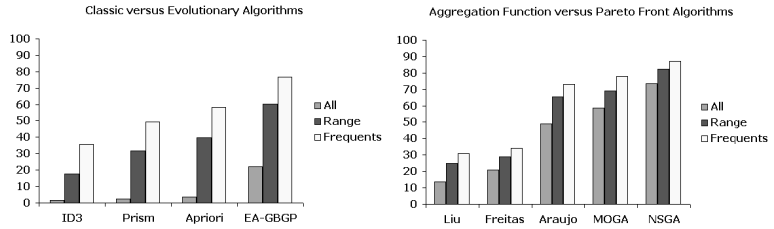


*Figure 14.* Percentage of Interesting Rules

In Figure 14 it is shown that evolutionary algorithms discover more interesting rules than classic algorithms. The best results are again obtained with algorithms based on Pareto Front and Araujo's Aggregation Function (this is because Araujo's contains a component related to interest).

**Percentage of Comprehensible Rules.** Figure 15 shows the percentage of comprehensible rules (rules with a value of simplicity greater than 0.5) discovered with classic versus evolutionary algorithms and different versions of GBGP.

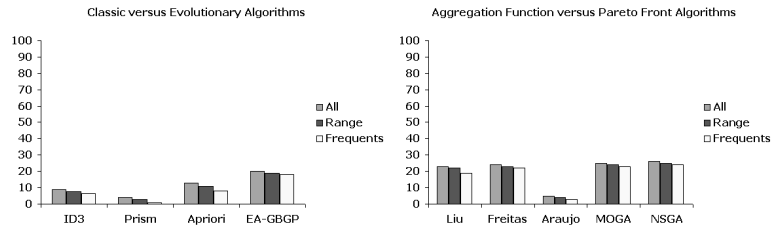


*Figure 15.* Percentage of Comprehensible Rules

Analyzing the results in Figure 15 we can see that evolutionary algorithms discover rules that are more comprehensible, especially, algorithms based on Pareto Front and Liu and Freitas Aggregation

Function (this is due to the fact that Liu and Freitas' contain a component related to simplicity).

Finally we are going to summarize the main results obtained from the previous comparisons. In general classic algorithms, and mainly the Apriori's, discover very accurate rules, but they fail to obtain short rules with high interestingness. Also, in the case of using a lot of data (all available data), they generate a number of rules so huge that it makes them impossible to be used later. In general evolutionary algorithms produce a smaller number of rules than classic algorithms and also, the percentage of comprehensible and interesting rules is significantly higher. Among the approaches based on aggregation functions, Liu and Freitas are centered fundamentally on optimizing the accuracy and the compressibility of the rules, and Alves on the interestingness of the rules. However the use of algorithms based on the concept of Pareto Front (MOGA and NSGA) can simultaneously optimize the three objectives, discovering the biggest percentage of exact, comprehensible and interesting rules.

— A final comment about the scalability of algorithms in terms of how fast is the rule discovery. Classic algorithms are fast when the size of data is small (frequent data) but they are extremely slow when the size of data grows (range data and all available data). On the contrary, evolutionary algorithms are more independent from the size of data, and their speed generating rules is less variable than classic algorithms. —

### 7.4. Description of Discovered Knowledge

— The author of the course has a fundamental role in this form of rule mining, because he can guide the search by setting subjective restrictions 11 using his own knowledge and experience in education. That is, using all available data, only frequent data or range data, using data about one specific chapter or full course, using data about all students or only students with an EXPERT or BEGINNER final knowledge level, using only data about times, levels and successes to compose the rules' antecedent or consequent, etc. —

It is important to mention that the rule comprehensibility and interestingness are subjective concepts, difficult to quantify effectively. Due to this, we have used constraint-based mining (Han et al., 1999), in which the user provides constraints that guide the search. We use three types of constraints:

1. Data constraints. The teacher can specify the relevant data set to the mining task.

2. Rule constraints. The teacher can select specific constraints on the rules to be mined.

3. Interestingness constraints. The teacher can set what ranges of a measure are interesting for him.

As we have commented before, our objective is to show a group of useful rules to the teacher so that he can take decisions about how to improve the course. From a semantic point of view, the pattern of discovered rules is:

**IF** Level|Time|Success **AND** ... **THEN** Level|Time|Success

*Where* :

**Level,Time and Success:** are level, time and sucess conditions.

The discovered rules show different types of relationships depending on what the attributes in the rule consequent are:

- **Time.** It shows which attributes (attributes in the rule antecedent) have an influence on the time (attribute of the rule consequent).

- **Level.** It shows which attributes (attributes in the rule antecedent) have an influence on the level (attribute of the rule consequent).

- **Success.** It shows which attributes (attributes in the rule antecedent) have an influence in the success (attribute of the rule consequent).

These relationships can make reference to chapters, to concepts or to concepts' scenarios of web-based adaptive courses. Using these discovered relations the teacher can make decisions about what modifications in the course are the most appropriate to increase the relationship (if he considers it to be desirable) or on the contrary to eliminate the relationship (if he considers it not to be desirable) by changing or modifying the contents, the structure or the adaptation of the course.

In general, we can discover tree types of rules depending on elements referred in the rule (chapters, concepts or scenarios of concepts):

**Rules about chapters.** They show relationships between different chapters of the same course. They refer to knowledge levels obtained in initial and final tests). The pattern of this type of discovered rules is:

**IF** ChapterLevel **AND** ... **THEN** ChapterLevel

*Where* :

**ChapterLevel:** are conditions about tests levels.

Using this information, the teacher can make decisions such as changing the sequence of chapters in the course, or joining them to form one only chapter if he wants to increase the relation, or on the other hand to put them further apart in the course if he wants to decrease the relation.

**Rules about concepts.** They show relationships between different concepts of the same or different chapter. They refer to knowledge levels obtained in activities. The pattern of this type of discovered rules is:

**IF** ConceptLevel **AND** ... **THEN** ConceptLevel
*Where* :
**ConceptLevel:** are conditions about activities levels.

Using this information, the teacher can make decisions such as to put activities in the same chapter (if they are in different chapters), to put them in the same difficulty level (if they have different difficulty levels), to put them sequentially in the chapter, or on the contrary to put them further apart within the chapter or in the course.

**Rules about scenarios of concepts.** They show relationships between scenarios of type exposition and/or activity. They refer to times, successes and levels obtained in exposition content pages and evaluation activities of concepts. The pattern of this type of discovered rules is:

**IF** ScenarioLevel|ScenarioTime|ScenarioSuccess **AND** ...
**THEN** ScenarioLevel|ScenarioTime|ScenarioSuccess
*Where* :
**ScenarioLevel,ScenarioTime,ScenarioSuccess:**
are conditions about scenarios times, successes and levels.

Using this information, the teacher can make decisions such as to delete pages because they are either duplicate, or badly phrased or they are incorrect; to change the difficulty level of the referred concept, and to modify the content and/or answers of questions since they are duplicate, badly phrased or incorrect, or to change the difficulty level of the referred concept.

As we can see, these rules can be used to improve adaptive web-based courses, although it could be used in other web-based systems and not only in the adaptive ones. But our aim is to improve courses with

adaptive functionalities like personalization, using different difficulty levels.

— These results would easily generalize to other courseware apart from AHA!(De Bra et. al., 2003), and EPRules tool (Romero et al., 2002) can be used only carrying out the following:

1. **Adaptation of the usage information.** The typical log information (Heift and Nicholson, 2000) stored by web-based system is only about times in accessed pages. In web-based eduction (Brusilovsky, 2001), students are evaluated using usually quiz and test, and obtained scores are stored. Finally, some adaptive applications (De Bra et. al., 2003) use these scores to obtain different knowledge levels. But if not, knowledge levels can be set manually using scores as usually done in traditional education when teacher marks exams. So, the usage information needed to do rule mining with EPRules can easily be obtained and adapted to our specific data format.

2. **Correspondence with the domain model.** The domain model in web-based educational system (Brusilovsky, 2001) is composed of a set of small domain knowledge elements (DKE). DKE concepts can be named differently in different systems: concepts, items, topics, etc. and can represent bigger or smaller pieces of domain knowledge. Concretely in our system there are concepts (with several scenarios) grouped in chapters (Romero et al., 2002). So, in order to do rule mining with EPRules, it is only necessary to do a correspondence between the two domain models.

—

Next we are going to describe the meaning and the possible use of several discovered rules using EPRules.

**IF** LEVEL.interface-network-high = EXPERT
**THEN** LEVEL.tcpip-telnet-medium = EXPERT
$(Interest = 0.57, FactorCertainty = 0.75, Simplicity = 1)$

This first rule shows that the knowledge levels obtained in the evaluation activities of different concepts have been simultaneously very high (EXPERT). This indicates that the concepts (NETWORK with a HIGH difficulty level in the INTERFACE chapter, and TELNET with a MEDIUM difficulty level in the TCPIP chapter) are related to each other. In this case, the teacher should check the presented content for both concepts and try to find the reason for the relationship. He should also decide if joining both concepts into a single concept, putting both concepts in the same chapter, setting them to the same

level of difficulty, correcting the rules that assign levels, or any other modification would be helpful. In this specific rule example we have considered that both concepts should have the same level of difficulty. But if the levels refer to an initial or final test instead of activities, it can be concluded that the chapters are related. Then the teacher can join the chapters, or put them one after the other, or on the contrary, create more distance between them. (We only *indicate* relationships but do not automatically generate suggestions on how to deal with them.)

**IF** $TIME.testf - administration - high(0) = HIGH$
**THEN** SUCCESS.testf-administration-high(0)=NO
$(Interest = 0.51, FactorCertainty = 0.79, Simplicity = 1)$

This second rule shows the relationship between the time used for reading a question (the question number 0 of the final test in HIGH grade in the Administration chapter) and the success or failure obtained in the answer of this question. This relationship indicates that the question is not well enunciated or it has some kind of error, since students not only need a HIGH time in reading it, but they then answer it incorrectly. When the teacher discovers this type of relationship he should correct it by modifying the enunciation of the question or by replacing it by another question. In this concrete example rule we have found the enunciation of the question corresponding to the ADMINISTRATION concept to be confusing, and we had to replace it by another (similar but clearer) question.

**IF** $LEVEL.emulators - program - high = EXPERT$
**THEN** SUCCESS.emulators-program-high(1)= NO
$(Interest = 0.69, FactorCertainty = 0.73, Simplicity = 1)$

This third rule shows the relationship between the knowledge level obtained by students in the activity for evaluating a concept (the concept EMULATORS that has a HIGH level of difficulty in the PROGRAMS chapter) and the success or failure in answering a certain question of this activity. This relationship indicates that if a question is answered incorrectly by a large number of EXPERT level students then the question may not be well enunciated. In this specific example rule we found question 1 about the concept EMULATORS to be confusing (in the enunciation of the question) and we have rewritten it to solve the problem.

## 8. Conclusions

In this article we have introduced a methodology to improve Adaptive Systems for Web-Based Education. This methodology uses evolutionary algorithms as a data mining method for discovering interesting relationships in students' usage data. After we have analyzed the different existing methods (Tan et al., 2002) (Lavrac et al., 1999) (Yao and Zhong, 1999) for evaluating the quality of the rules obtained by knowledge discovery algorithms, we have determined the necessity of using multi-objective algorithms instead of classic algorithms (mono-objective algorithms). We have proposed the use of evolutionary approaches based on aggregation functions and Pareto Front. The comparison of algorithms with respect to the number of obtained rules and the percentage of interesting, accurate and comprehensible rules, shows that the algorithms based on Pareto Front (MOGA and especially NSGA) are superior compared to the other proposed algorithms, that use one only evaluation measure or a composition of several measures. With regard to the utility of the discovered rules to help in making decisions about possible modifications that can be carried out in ASWEs, we have described the different types of obtained rules, the utility of each type for the improvement of the course and and we have given specific examples of discovered rules obtained in the Linux course developed with AHA!. Finally, we have developed a specific tool in order to facilitate the realization of the full process of knowledge discovery by a non expert person in data mining. This tool allows to carry out the pre-processing of students' usage data, to place restrictions on the types of relationship we want to discover, as well as the application of the data mining algorithms for the rule extraction and the visualization of these rules.

The main conclusions we have obtained after developing the current work are the following:

1. We have shown that the use of data mining techniques (in our case prediction rule mining) on the usage information generated by ASWEs, allows to discover useful knowledge to improve the systems. This knowledge in the form of prediction rules can be used by the teacher to make decisions about how to modify the course to improve system performance.

2. We have proposed a methodology to improve ASWEs. But this methodology can be applied to other types of web-based systems because it is domain independent. The only difference is the type of usage information of each particular system.

3. We have developed our own ASWE using AHA! and a specific web usage mining tool in order to implement the proposed methodology completely. In this way, a teacher can easily apply our methodology.

4. We have shown that evolutionary algorithms, and more specifically grammar based genetic programming algorithms, are very suitable for rule discovery in ASWE. This is not only due to the ability to obtain more interesting and comprehensible rules but also to its flexible capacity to represent the individuals. The teacher can change the individuals' format by only modifying the rule codification.

5. We have shown that the use of an evolutionary multi-objective approach can improve the obtained results. Concretely, the NSGA approach obtains a smaller number of rules with greater interest, accuracy and simplicity than the other algorithms used.

6. We have shown that the discovered rules over the usage data of a Linux course using specific teacher restrictions in the rules, are interesting, coherent in most of the cases, and can be used to improve questions, concepts, etc. of the course.

As a current line of research we are beginning to work on the search for new measures related to the rules' subjective interest using professionals in education. Thus, they have to provide an interest value for each of the discovered rules. The teachers will have to decide which are the most interesting rules, in an interactive way, during the process of rule discovery. In this same research line, there are some references to the evolutionary hot spots miner (Williams, 1999) in which the individuals are directly evaluated by an expert in each cycle of the algorithm. Perhaps this method may not be applicable to our problem due to the large number of rules that can be obtained in each evolution. However, a first approach with a small size of population may be viable and could show information about measures that model in an effective way these user preferences in the discovered rules.

## Appendix

### A. Rule Evaluation Measures

At present there are a lot of rule evaluation measures (Tan et al., 2002) (Lavrac et al., 1999) that come from different areas such as machine learning, data mining, statistics, classification, etc. But almost all of them can be obtained from the contingency table. The contingency

table (Yao and Zhong, 1999) is the generalization of the confusion matrix that is used for the rule evaluation in classification problems. The contingency table of the generic rule $A \rightarrow C$, where A is the antecedent of the rule, and C is the consequence of the rule, is shown in Table VI.

Table VI. Contingency Table

| | A | $\neg A$ | |
|---|---|---|---|
| C | n(AC) | n( $\neg A$C) | n(C) |
| $\neg C$ | n(A $\neg C$) | n($\neg A \neg C$) | n($\neg C$) |
| | n(A) | n(A) | N |

**A:** instances that match the rule antecedent affirmation.

$\neg A$ : instances that matches the rule antecedent negation.

**C:** instances that match the rule consequent affirmation. ($\neg C$ is similar but in negation).

**AC:** intersection of A and C. ($\neg AC, \neg A \neg C$ and $A \neg C$ are similar).

**n(A):** cardinality of A. Number of instances of A. ($n(C), n(\neg A)$ and $n(\neg C)$ are similar).

**N:** the total number of data instances.

We have also used the following probabilities:

**p(A):** relative frequency of A, obtained by $p(A) = \frac{n(A)}{N}$ ($p(C), p(\neg A)$ and $p(\neg C)$ are similar).

**p(AC):** relative frequency of the intersection of A and C, obtained by $p(AC) = \frac{n(AC)}{N}$ ($p(\neg AC), p(\neg A \neg C)$ and $p(A \neg C$ are similar).

**p(A/C):** relative frequency of A conditioned by C, obtained by $p(A/C) = \frac{p(AC)}{p(C)}$ (it is similar to $p(\neg A/C), p(\neg A/\neg C)$ and $p(A/\neg C)$.

Several of the most used rule evaluation measures (Tan et al., 2002) (Lavrac et al., 1999) are shown in Table VII:

Table VII. Rule Evaluation Measures.

| Name | Expression |
|---|---|
| Support | $Sup(A \rightarrow C) = p(CA) = \frac{n(CA)}{N}$ |
| Confidence | $Conf(A \rightarrow C) = p(C/A) = \frac{p(CA)}{p(A)}$ |
| Laplace | $Lap(A \rightarrow C) = \frac{Sup(A \rightarrow C)+1}{Sup(A)+2}$ |
| Conviction | $Conv(A \rightarrow C) = \frac{p(A)p(\neg C)}{p(A\neg C)}$ |
| Interest | $I(A \rightarrow C) = \frac{p(CA)}{p(C)p(A)}$ |
| P-S Interestingness | $RI(A \rightarrow C) = p(CA) - p(C) * p(A)$ |
| T-K Interestingness | $IS(A \rightarrow C) = \sqrt{I(A \rightarrow C) * \frac{p(CA)}{N}}$ |
| Klsgen | $K(A \rightarrow C) = p(A)^{\alpha} * (P(C/A) - p(C))$ |
| Leverage | $Lev(A \rightarrow C) = p(C/A) - (p(A) * p(C))$ |
| Quinlan | $Q(A \rightarrow C) = \frac{n(CA)-1/2}{n(A)}$ |
| Chi-squared | $\chi^2(A \rightarrow C) = \frac{N(n(AC)*n(\neg A\neg C)-n(A\neg C)*n(\neg AC))^2}{n(A)*(\neg A)*n(C)*n(\neg C)}$ |
| Correlation coefficient | $\rho(A \rightarrow C) = \frac{n(AC)*n(\neg A\neg C)-n(\neg AC)*n(A\neg C)}{\sqrt{n(A)*(\neg A)*n(C)*n(\neg C)}}$ |
| Predictive Association | $\lambda(A \rightarrow C) = \frac{\sum_j max_k n(C_k,A_j) - max_k n(C_k)}{N - max_k * n(C_k)}$ |
| Entropy | $H(A \rightarrow C) = -\sum_j \sum_l p(A_k B_j * log_2 p(A_k B_j))$ |
| Certainty Factor | $CF(A \rightarrow C) = max(\frac{P(C/A)-P(C)}{1-p(C)}, \frac{P(A/C)-P(A)}{1-p(A)})$ |
| Gini | $Gini(A \rightarrow C) = p(A) * P(\frac{C}{A})^2 + p(\neg\frac{C}{A})^2 + p(\neg A) * p(C/\neg A)^2 + p(\neg C/\neg A)^2 - p(C)^2 - p(\neg C)^2$ |
| Gain Function | $Gain(A \rightarrow C) = p(AC) - \Theta * p(A)$ |
| J-measure | $J(A \rightarrow C) = p(C) * (p(A/C) * log_2(\frac{p(A/C)}{p(A)}) + (1 - p(A/C)) * log_2(\frac{1-p(A/C)}{1-p(A)}))$ |
| Divergence | $H(A \rightarrow C) = p(A) * (\frac{p(CA)}{p(A)} * log_2(\frac{p(CA)/p(A)}{p(C)}) + \frac{p(\neg CA)}{p(A)} * log_2(\frac{p(\neg CA)/p(A))}{p(\neg C)}))$ |
| Negative Reliability | $NegRel(A \rightarrow C) = p(\neg C/\neg A)$ |
| Sensitivity | $Sens(A \rightarrow C) = p(A/C)$ |
| Specificity | $Spec(A \rightarrow C) = p(\neg A/\neg C)$ |
| Coverage | $Cov(A \rightarrow C) = p(A)$ |
| Novelty | $Nov(A \rightarrow C) = p(CA) - p(C) * p(A)$ |
| Satisfaction | $Sat(A \rightarrow C) = \frac{p(\neg C)-p(\neg C/A)}{p(\neg C)}$ |
| Informativity | $Inf(A \rightarrow C) = -log_2(p(C/A))$ |
| Relative Accuracy | $RAcc(A \rightarrow C) = p(C/A) - p(C)$ |
| Weighted RAcc | $WRAcc(A \rightarrow C) = p(A) * (p(C/A) - p(C))$ |
| Necessity | $N(A \rightarrow C) = \frac{p(\neg A/C)}{p(\neg A/\neg C)}$ |
| Characteristic Interest | $IC(A \rightarrow C) = 1 - N(A \rightarrow C) * p(C)$ |
| Significance | $Sig(A \rightarrow C) = P(C/A) * log_2(I(A \rightarrow C))$ |
| Relative Risk | $R(A \rightarrow C) = \frac{p(C/A)}{p(C/\neg A)}$ |
| Simplicity | $Simp(A \rightarrow C) = (1 - \frac{AntecedentSize}{MaximumSize})$ |

## Acknowledgements

The authors* gratefully acknowledge the financial support provided by the Spanish Department of Research of the Ministry of Science and Technology under TIC2002-04036-C05-02 Projects.

## References

Agrawal, R. Imielinski, T. Swami, A.: 1993, Mining association rules between sets of items in large databases. In: *Proc. of the 1993 ACM SIGMOD International Conference on Management of Data. Washington, D.C., pp.207–216.*

Araujo, D.L.A. Lopes, H.S. Freitas, A.A.: 1999, A Parallel Genetic Algorithm for Rule Discovery in Large Databases. In: *Proc. Conf. IEEE Systems and Cybernetics. Tokyo, pp.940–945.*

Arruabarrena, R. Prez, T.A. Lpez-Cuadrado, J. Gutirrez, J.: 2002, On Evaluating Adaptive Systems for Education. In: *Second Conf. AH2002. Adaptive Hypermedia and Adaptive Web-based Systems. Nicosia, Cyprus, pp.363–367.*

Askira-Gelman, I.: 1998, Knowledge Discovery: Comprehensibility of the Results. In: *Hawaii Int. Conf. on System Sciences. Kohala Coast, HI, pp.247–255.*

Bayardo, R. J. Agrawal, R.: 1999, Mining the most interesting rules. In: *Fifth Conf. ACM on Knowledge Discovery and Data Mining SIGKDD ,San Diego, CA, USA, pp.145–154.*

Bojarczuk, C.C. Lopes, H.S. Freitas, A.A.: 2001, Constrained-syntax genetic programming for knowledge discovery in medical databases. In: *10th Int. Symposium on System Modeling ,Zakopane, Poland.*

Cendrowska, J.: 1987, PRISM: an algorithm for inducing modular rules. *Journal of Man-Machine Studies* **27**, 349–370.

Delgado, M. Snchez, D. Martn-Bautista, M.J. Vila, M.A.: 2001, Mining Association rules with improved semantics in medical databases. *Artificial Intelligence in Medicine* **21**, 241–245.

Dhar, V. Chou, D. Provost, F.: 2000, Discovering Interesting Patterns for Investment Decision Making with GLOWER. *Data Mining and Knowledge Discovery* **4**, 251–280.

Dougherty, J. Kohavi, M. Sahami, M.: 1995, Supervised and unsupervised discretization of continuous features. *In:* Int. Conf. Machine Learning Tahoe City, CA, pp.194–202.

Minaei-Bidgoli, B. Punch III, W.F.: 2003, Using Genetic Algorithms for Data Mining Optimization in an Educational Web-based System.. In: *Genetic and Evolutionary Computation Conference. Chicago, Illinois, USA, pp.2252–2263.*

Brusilovsky, P.:1998, Adaptive Educational Systems on the World-Wide-Web: A Review of Available Technologies. In: *Int. Conf. on Intelligent Tutoring Systems*, San Antonio, TX.

Brusilovsky, P.: 2001, Adaptive Educational Hypermedia. In: *Proc. of Tenth Int. PEG Conference*, Tampere, Finland, pp. 8–12.

Carro, R.M. Pulido, E. Rodriguez, P.: 1999, Desinging Adaptive Web-based Courses with TANGOW. In: *Conf. Computers in Education*, Chiba, Japan, pp.697-704.

Coello, C.A Veldhuizen, D.A. Lamount, G.B.: 2002, Evolutionary Algorithms for Solving Multi-Objective Problems.. *Kluwer.*

Darwin, C.: 1859, On the Origin of Species by Means of Natural Selection. *John Murray.*

Deb, K.: 2001, Multi-Objetive Optimization Using Evolutionary Algorithms. *Wiley.*

De Bra, P. Wu, H. Aerst, A. Houben, G.: 2000, Adaptation Control in Adaptive Hypermedia Systems. In: *Proc. of Int. Conference on Adaptive Hypermedia and Adaptive Web-based Systems*, Trento, Italy, pp. 250-259.

De Bra, P., Aerts, A., Berden, B., De Lange, B., Rousseau, B., Santic, T., Smits, D., Stash, N.: 2003, AHA! The Adaptive Hypermedia Architecture. In: *Proc. of the ACM Hypertext Conference*, Nottingham, UK, pp. 81–84.

De Castro, C. and Romero, C.: 2002, HAMTUTOR. Autor tool to develop adaptive multimedia courses. In: *World Conf. on E-Learning in Corporate, Government, Healthcare, Higher Education*, Montreal, Canada, pp. 2575–2576

Dubois, P.: 2002, Edicin Especial MySQL. *Prentice Hall.*

Fonseca, C.M. Fleming, P.J.: 1993, Genetic algorithms for multiobjective optimization: formulation, discussion and generalization. In: *Proc. 5rd Int. Conf. on Genetic Algorithms*, San Mateo, California, pp. 416–423.

Freitas, A. A.: 1997 A Genetic Programming Framework for Two Data Mining Tasks: Classification and Generalized Rule Induction. In: *Conf. Genetic Programming*, CA, USA, pp. 96–101.

Freitas, A. A.: 2000, Understanding the Crucial Differences Between Classification and Discovery of Association Rules. *ACM SIGKDD Explorations*, **2**(1), 65–69.

Freitas, A. A.: 2001, A Survey of Evolutionary Algorithms for Data Mining and Knowledge Discovery. In: Ghosh, A.; Tsutsui, S. (Eds.) *Advances in Evolutionary Computation, Springer-Verlag*, pp.819–845

Freitas, A. A.: 2002, Data Mining and Knowledge Discovery with Evolutionary Algorithms. *Springer-Verlag.*

Gilbert, R.G. Goodacre, R. Shann, B. Kell, D.B. Taylor, J. Rowland, J.J.: 1998, Genetic Programming-based variable selection for high-dimensional data. In: *Proc. 3rd Conf. Genetic Programming*, San Francisco, CA, USA, pp.109–115.

Ghosh, A. Nath B.: 2004, Multi-objective rule mining using genetic algorithms. *Information Sciences*, **163**, 123–133.

Han, J. Lakshamanan, L. Raymond, T.Ng.: 1999, Constraint-Based, Multidimensional Data Mining. *IEEE Computer*, **32** (8), 46–50.

Heift, T. Nicholson, D.: 2000, Enhanced Server Logs for Intelligent, Adaptive Web-based Systems. *Technical Report. Institute for Semantic Information Processing. Universitt Osmabrk.*

Herin, D. Sala, M. Pompidor, P.: 2002, Evaluating and Revising Courses from Web Resources Educational. In: *Int. Conf. on Intelligent Tutoring Systems*, Biarritz, France, San Sebastian, Spain, pp. 208–218.

Hipp, J. Gntzer, U. Nakhaeizadeh, G.: 2000, Mining Association Rules: Deriving a Superior Algorithm by Analyzing Today's Approaches. In: *European Symposium Data Mining and Knowledge Discovery*, Lyon, France, pp. 13–16

Koza, J.R.: 1992, Genetic Programming: on the programming of computers by means of natural selection. *MIT Press.*

Lavrac, N. Flach, P. Zupan B.: 1999, Rule Evaluation Measures: A Unifying View. In: *Int. Workshop on Inductive Logic Programming*, Springer-Verlag, pp. 174–185.

Liu, B. Hsu, W. Chen, S. Ma, Y.: 2000, Analyzing the Subjective Interestingness of Association Rules. *IEEE Intelligent Systems,* **15**(5), 47–55.

Liu, J.J. Kwok, J.T.: 2000, An Extended Genetic Rule Induction Algorithm. In: *Proc. of the Congress on Evolutionary Computation,* La Jolla, California, USA, pp.458–463.

Michalski, Z.: 1998, Genetic Algorithms + Data Structures = Evolution Program. *Springer..*

Mitra, S. Pal S.K. Mitra, P.: 2001, Data Mining in Soft Computing Framework: A Survey. *IEEE Transaction on Neural Networks*, **13**(1), 3–14.

Noda, E. Freitas, A. Lopes, H.S.: 1999, Discovering interesting prediction rules with a genetic algorithm. In: *Proc. Congress on Evolutionary Computation,* Washington D.C., USA, pp.1322–1329.

Nordin, P. Banzhaf, W. Keller, R.E. Francone, F.D.: 1998, Genetic Programming: An Introduction. *Morgan Kaufmann.*

Ortigosa, A. Carro, R.M.: 2002, Asistiendo el Proceso de Mejora Continua de Cursos Adaptativos. In: *III Congreso Int. de Interaccin Persona-Ordenador,* Granada, pp.246–250.

Pahl, C. Donnellan, D.: 2002, Data Mining Technology for the Evaluation of Web-based Teaching and Learning Systems. In: *Proc. Congress E-learning,* Montreal, Canada.

Piatesky-Shapiro, G Matheus, J.: 1994, The interestingness of deviations. In: *AAAI Workshop on Knowledge Discovery in Databases,*Seattle, Washington, pp.25–36.

Pierrakos, D. Paliouras, G. Papatheodorou, C. Spyropoulos C.D.: 2003, Web Usage Mining as a Tool for Personalization: A Survey. *User Modeling and User-Adapted Interaction.*, **12**(4), 311–371.

Quilan, J.R.: 1987, Generating Production rules from decision trees. In. *Proc. of IJCAI*, pp. 304–307.

Klösgen, W. Zytkow, J.M.: 2002, Handbook of Data Mining and Knowledge Discovery. *Oxford University Press.*

Ratle, A. Sebag, M.: 2000, Genetic Programming and domain knowledge: beyond the limitations of grammar guided machine discovery. In. *Proc. of 6th Conf. Parallel proglem solving for nature,* Paris, France, pp.211–220.

Romero, C. De Bra, P. Ventura, S. De Castro, C.: 2002, Using Knowledge Level with AHA! For Discovering Interesting Relationship. In: *Proc. of the AACE ELearn'2002*, Montreal, Canada, pp. 2721–2722.

Romero, C. Ventura, S. De Bra, P. De Castro, C.: 2002, Discovering Prediction Rules in AHA! Courses. In: *9th Int. Conf. on User Modeling*, Johnstown, PA, USA, pp.25–34.

Ryu, T.W. Eick, C.F.: 1996, Discovering Commonalities of a set of Objects Usign Genetic Programming. *Proc. of Genetic Programming Conference.*

Sarawagi, S. Thomas, S. Agrawal, R.: 1998, Integrating Association Rule Mining with Relational Database Systems: Alternatives and Implications. In: *Conf. on Management of data,* Seattle, Washington, pp.343–354.

Shortliffe, E. Buchanan, B.: 1975, A model of inexact reasoning in medicine. *Mathematical Biosciences*, **23**, 351–379.

Smythe, P. Goodman, R.M.: 1992, An Information Theory Approach to Rule Induction from Databases. *IEEE Knowledge and Data Engineering.* **4**(4), 301–316.

Silberschatz, A.: 1995. On Subjective Measures of Interestingness in Knowledge. In: *Proc. Knowledge Discovery and Data Mining*, Montreal, Canada, pp. 275–281.

Silverstein, A. Brin, S. Motwani, R.: 1998, Beyond market baskets : Generalizing association rules to dependence rules. *Data Mining and Knowledge Discoverty.*, **2**, 29–68.

Spiliopoulou, M.: 2000, Web Usage Mining for Web Site Evaluation.. *Communicacion of the ACM,* **43**(8) 127–134.

Srinivas, N. Deb, K.: 1994, Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation,* **2**(3), 217–249.

Srivastava, J. Cooley, R. Deshpnade, M. Tan, P.: 2000, Web Usage Mining: Discovery and Applications of Usage Pattern from Web Data. *ACM SIGKDD,* **1**(2):12–23.

Tan, P. Kumar, V. Srivastava, J.: 2002, Selecting the right Interestingness measures for association patterns. In: *Proc. of the 8th Int. Conf. on Knowledge Discovery and Data Mining,* Edmonton, Canada, pp.32–41.

Tang, T. McCalla, G.: 2002, Student Modeling for a Web-based Learning Environment: a Data Mining Approach. In: *Proc. Conf. on Artificial Intelligence AAAI,* Edmonton, Alberta, Canada pp.967–968.

Ventura, S. Ortiz, D. Hervz, C.: 2002, Jclec: Una biblioteca de clases java para computacin evolutiva. In: *I Congreso Espaol de Algoritmos Evolutivos y Bioinspirados,* Merida, pp.23–30.

Wang, F.: 2002, On Using Data-Mining Technology for Browsing Log File Anlisis in Asynchronous Learning Environment. In: *Conf. on Educational Multimedia, Hypermedia and Telecomunications,* Denver, Colorado, pp.2005–2006.

Whigham, P.A.: 1995, Gramatically-based Genetic Programing. In: *Proc. of the Workshop on Genetic Programming,* California, pp.33–41.

Williams, G.J.: 1999, Evolutionary Hot Spots Data Mining. An Architecture for Exploring for Interesting Discoveries. In: *Conf. on Knowledfe Discovery and Data Mining,* Beijing, China, pp. 184–193.

Witten, I.H. Frank, E.: 2000, Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. *Morgan Kaufmann.*

Wong, M.L. Leung, K.S.: 2000, Data Mining using Grammar Based Genetic Programming and Applications. *Kluwer.*

Yao, Y. Zhong, N.: 1999, An Analysis of Quantitative Measures Associated with Rules. In: *Proc. of PAKDD'99,* pp.479–488.

Yu, P. Own, C. Lin, L.: 2001, On Learning Behavior Analysis of Web Based Interactive Environment. In: *Proc. ICCEE*, Oslo/Bergen Norway.

Zaïane, O.R. Luo, J.: 2001, Towards Evaluating Learners' Behaviour in a Web-Based Distance Learning Environment. In: *Proc. IEEE Int. Conf. on Advanced Learning Technologies,* Madison, Wisconsin, pp.357–360.

Zaïane, O.R.: 2002, Building a Recommender Agent for e-Learning Systems. In: *Proc. Int. Conf. on Computers in Education,* Auckland, New Zealand, pp.55–59.

**Vitae**

**Dr. Cristóbal Romero** is Assistant Professor in the Computer Science Department of the Cordoba University (Spain). He received his ph. d. in Computer Science from the University of Granada in 2003. His research interests are in artificial intelligence in education.

**Prof. dr. Sebastián Ventura** is Professor in the Computer Science Department of the Cordoba University (Spain). He received his ph. d. in Sciences from the University of Cordoba in 1996. His research interests are in evolutionary computation and web-based systems.

**Prof. dr. Paul De Bra** is Professor in the Computer Science Department of the Eindhoven University of Technology (Netherlands). He received his ph. d. in Computer Science from the University of Antwerp in 1987. His research interests are in adaptive hypermedia systems and web-based information systems.