User Modeling and User-Adapted Interaction (2004) 14: 425–464 © Springer 2005
DOI 10.1007/s11257-004-7961-2

Knowledge Discovery with Genetic Programming for Providing Feedback to Courseware Authors

CRISTÓBAL ROMERO[1], SEBASTIÁN VENTURA[1] and PAUL DE BRA[2]
[1]*Department of Computer Sciences and Numerical Analysis, University of Córdoba, 14071 Córdoba, Spain. e-mail: {cromero, sventura}@uco.es*
[2]*Department of Computer Sciences, Eindhoven University of Technology, PO Box 513, 5600 MB Eindhoven, Netherlands. e-mail: debra@win.tue.nl*

(Received: 26 February 2004; accepted in revised form: 11 December 2004)

Abstract. We introduce 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. Such knowledge may be very useful for teachers and course authors to select the most appropriate modifications to improve the effectiveness of the course. We use Grammar-Based Genetic Programming (GBGP) with multi-objective optimization techniques to discover prediction rules. We present a specific data mining tool that can help non-experts in data mining carry out the complete rule discovery process, and demonstrate its utility by applying it to an adaptive Linux course that we developed.

Key words. adaptive system for web-based education, data mining, evolutionary algorithms, grammar-based genetic programming, prediction rules

1. Introduction

Web-based education has considerably gained in importance, spurred by the fact that neither students nor teachers are bound to a specific location and that this form of computer-aided instruction is virtually independent of any specific hardware platform (Brusilovsky, 2001). Thousands of web courses have been deployed in the past few years. However, most of them are merely a network of static web pages. This led to orientation and comprehension problems in students since navigation in such courses is completely unrestricted. Adaptive Systems for Web-based Education (ASWE) provide a superior alternative (Brusilovsky, 1998). They are the result of a joint evolution of Intelligent Tutoring Systems and Adaptive Hypermedia Systems and have the most desirable characteristics of both, namely, they increase students' interaction with the educational system and adapt to the needs of each individual student.

The development of an ASWE is a laborious activity (Hérin et al., 2002), and it becomes even more complex when the number of adaptation possibilities increases. 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. Due to the complexity of these decisions, a one-shot design is hardly feasible, even when it is carefully done. Instead, it will be necessary in most cases to evaluate and possibly redesign the ASWE based on students' usage information, preferably even in a continuous manner (Ortigosa and Carro, 2002). To facilitate this we need methods and tools to observe students' behavior and to assist teachers in detecting possible errors, shortcomings and possible improvements. A very promising area for such methods and tool is data mining.

In the past few years, researchers have begun to investigate various knowledge discovery and data mining methods (Zaïane and Luo, 2001) to help teachers validate ASWEs. These methods allow one to discover new knowledge based on students' usage data. The idea has already been successfully applied in e-commerce systems where its use is now very popular (Spiliopoulou, 2000). Comparatively little work in this direction has been conducted in Web-based Education yet. This is the line of research that will be introduced in this paper.

Data mining (DM) is a multidisciplinary area in which several computing paradigms converge: decision tree construction, rule induction, artificial neural networks, instance-based learning, Bayesian learning, logic programming, statistical algorithms, etc. (Klösgen and Zytkow, 2002). The objective of data mining is to discover new interesting and useful knowledge. Evolutionary Algorithms (EAs) are one of the newer methods for this purpose. The main advantages of EAs over the classical greedy induction algorithms lie in their ability to perform a global search and the way they treat the attribute interaction problem (Freitas, 2002). Evolutionary algorithms have been inspired by Darwinian evolution (Darwin, 1859), where each individual codifies a solution and evolves to a better individual by means of genetic operators (mutation and crossover). Hence, evolutionary algorithms perform a search in the space of the candidate individuals in a similar manner as induction algorithms. The difference lies in the search strategy used. Classical learning induction algorithms normally use a local greedy search strategy, while evolutionary algorithms use a global search strategy. 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 should always be interesting, novel and useful for the end-user (Bayardo and Agrawal, 1999). While data mining algorithms usually discover comprehensible and exact rules, these are often not interesting. "Interest" is also difficult to operationalize. Two types of methods exist to select interesting rules, namely subjective and objective methods (Liu et al., 2000). Subjective methods are domain-dependent and user-directed while objective methods are data-directed, domain-independent and use some evaluation measure. A large number of evaluation measures such as confidence, support, interest, gini, laplace, etc. have been described in the literature Lavrac et al. 1999; Tan et al. 2002; Yao and Zhong 1999; see Appendix A). But each of these measures is focused on a specific aspect of the discovered rules and there is no

measure that is significantly better than the others in every application domain. For this reason, it seems necessary to consider several measures simultaneously. A simple way is to combine the weighted individual measures into a single metric. However, this is not 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 a 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 is not 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). In this paper, we are going to use this approach.

We divided this paper into the following sections. We first review the background research related to the use of data mining in adaptive educational systems in Section 2. We then describe the specific knowledge discovery problem that we want to resolve and our proposed methodology in Section 3. In Section 4, we discuss the collection and preparation of usage data, using an ASWE that we developed as a concrete example. In Section 5, we present the evolutionary algorithms for rule discovery that we developed using multi-objective genetic programming. Section 6 gives an overview of the rule discovery and visualization tool that we developed, presents the experimental results of the different tests that we performed, and describes rules that we discovered in our example ASWEs. Finally, Section 7 summarizes the main results and conclusions, and indicates areas of future research.

2. Related Work

The need to analyze the vast amount of information that is generated daily on the web has spawned a considerable interest in web data mining. Web data mining is the application of data mining methods to web data (Srivastava et al., 2000). Three different types can be distinguished depending on the data used:

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

Web Structure Mining mines the structure of web hyperlinks, which represent the structure of web sites.

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

The major application areas of web usage mining are personalization, business intelligence, usage characterization, systems improvement and site redesign (Srivastava et al., 2000). The best developed applications are e-commerce systems which try to understand clients' interests to increase online sales (Spiliopoulou, 2000).

A more recent application of web usage mining is personalization (Pierrakos et al., 2003), which enables information systems to adapt to users' individual needs. A special case is the deployment of web mining to personalize educational systems. The use of data mining methods in e-learning systems can provide useful information to evaluate and improve these systems. Although this research area is still in its infancy, web-based educational systems that exploit the advantages of knowledge acquisition are already being developed (Zaïane 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 perspective, there are no differences since the objective of web mining in both application areas is to study users' behavior (namely of clients in e-commerce and students in e-learning systems), evaluate this behavior, and improve the systems to help the users. But from a user's point of view there are differences, because the e-commerce objective is to guide clients in purchasing while the e-learning objective is to guide students in learning. So, each of them has special characteristics that require a different treatment of the web-mining problem.

Web mining in education is not new but was already successfully employed in several web-based educational systems. Usually such applications consist in searching users' browsing patterns with one or more of the following algorithms:

- *Association rule mining and sequential pattern analysis.* These methods search for associations among visited web pages, and analyze sequences of pages that a user hits in a single visit or subsequent visits. In their pioneering article, (Zaïane and Luo, 2001) propose the discovery of useful patterns based on restrictions, to help educators evaluate students' activities in web courses. They also use recommended agents for e-learning systems using association rule mining (Zaïane, 2002), 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 logs, and apply fuzzy association rules to find out the relationships between each pattern of learner's behavior; included time spent on-line, numbers of articles read, number of articles published, number of questions asked, etc. Related research has been carried out by (Pahl and Donnellan, 2002). Their work is based on analyzing each student's individual sessions. They first define the learning period (of time) of each student and then split web server log files into individual sessions, calculate session statistics, and search for session patterns and time series. A third important piece of work is the one by (Wang, 2002), who uses associative material clusters and sequences among them. This knowledge allows teachers to study the dynamic browsing structure and to identify interesting or unexpected learning patterns. To do this, he discovers two types of relations: association relations and sequence relations between documents.
- *Clustering and classification.* These methods group users by navigation behavior, group similar navigation behaviors, etc. This approach was developed by

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 characteristics based on the sequence and the contents of the pages they visited. (Minaei-Bidgoli and Punch, 2003) classify students based on features extracted from the logged data, to predict their final grades. They use genetic algorithms to optimize a combination of multiple classifiers by weighing feature vectors.

All these current approaches use the visited pages as input to the search, and hence the discovered information describes relations between *pages*. In contrast, our proposed method also searches for relations between concepts and chapter units of web-based courses, and not only between pages. As we describe in more detail in the next section, we aim at discovering relations to not only restructure 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 Web-based Educational System is arduous and laborious (Carro et al., 1999), due to the fact that the courseware author must make important decisions regarding the following questions.

- How many chapters or lessons does a course contain, and which are the most appropriate for each different knowledge level distinguished by the system (e.g. expert, intermediate and beginning students)? What is the most adequate organization (navigation scheme) of these chapters?
- How many concepts does a chapter include, and which ones are the most appropriate for each student's knowledge level? What is the most adequate organization of these concepts?
- Which activities will be used to evaluate each concept and each chapter for each student's knowledge level? What is the most adequate organization of these activities?

Due to all these decisions, it is very difficult to determine the most suitable course structure, contents and activities for AWESs. In fact, it is very likely that different authors would propose different curricula for the same course. Another problem is the subjectivity when dividing the course into difficulty or accessibility levels. These inherent difficulties in "getting ASWEs right in the first place" speak for the merits of a different approach, namely to evaluate systems that have been already developed based on students' usage data, and to modify them based on the resulting findings.

The evaluation of educational systems like ASWEs is a process of data gathering to determine the value of the instruction. Currently there exist three main evaluation paradigms (Arruabarrena et al., 2002), namely formative, summative

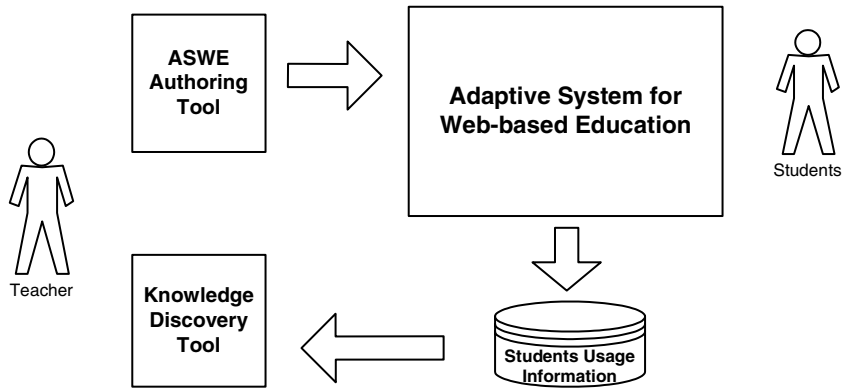


Figure 1. Proposed methodology to improve ASWEs.

and integrative evaluation. Three kinds of evaluation are commonly distinguished, namely internal, external and global evaluation. Finally, ASWEs also lend themselves to special evaluation techniques, including

comparison, to evaluate the characteristics of the system vis-a-vis other systems (that may be standards);
contact with users, to collect data about users' behavior and attitude;
data analysis, for reviewing and assessing collected data about user's interaction; and
pilot testing, to study the performance of the system with potential end-users.

But teachers usually evaluate a course from a student's point of view and focus on evaluating students' learning based on the scores they obtained. The course itself is usually not modified after being published on the web, and if so, modifications are only based on teachers' judgment or on the scores that students obtained when using it (for example, modifications related to questions that many students have failed). A deeper evaluation of students' interaction needs to be carried out though to improve a course.

Our suggested evaluation approach is based on data mining methods applied to students' usage data. As mentioned above, the originality of this work lies in the type of knowledge that we aim to discover. Previous research that we described above (Pahl and Donnellan, 2002; Tang and McCalla, 2002; Wang, 2002; Zaïane and Luo, 2001) aims at discovering associations between visited pages and thereby analyzes sequences of pages, groups browsing patterns and/or students, etc. Our aim is different though since we want to discover dependence relationships among elements and not among pages. Such elements can be concepts, chapters, questions, activities, etc. that are related to one or several different web pages in the course. We propose a development methodology that enables us to evaluate and improve ASWEs. Within this methodology, we added a specific evaluation step by using data mining techniques.

The proposed methodology consists of four main steps (also see Figure 1):

1. **Development of the ASWE.** The teacher builds the Hypermedia Adaptive Course, thereby providing the required information about the domain model, the pedagogical model and the interface module (the tutor model is usually constructed by the system itself and the data of the student model is acquired during execution time). To this end, the teacher normally uses an authoring tool, such as a general-purpose system like DreamWeaver, Toolbook and Director, or a special-purpose authoring tool such as AHA! (De Bra et al., 2003), HamWeb (De Castro and Romero, 2002), etc. After the course is completed the ASWE becomes published on a web server.
2. **Use of the ASWE.** Students log in and navigate in the course using a web browser. All information about system usage (times, levels and scores) is collected in the background and is stored in students' web server log files.
3. **Knowledge Discovery.** Students' log files are preprocessed and transferred to a database. Then knowledge discovery algorithms must be applied to discover important relationships among the collected data (in our case, we want to discover rules that relate students knowledge levels, times and scores). This can be done using a generic data mining tool like Weka (Witten and Frank, 2000), or a specific tool like EPRules (Romero et al., 2002) that we developed. In EPRules, a rule discovery algorithm must be selected as well as its parameters, and both the objective and subjective restrictions that the discovered rules should fulfill. After the algorithm terminates, the discovered rules are displayed, specifically the elements of the rule antecedents and consequents as well as the evaluation measures of each rule. The teacher then has to determine whether the group of discovered rules is interesting or not, and which rules should be used to advise on course modifications. This depends on their number, their quality with respect to the different measures, and their semantic meaning. If the rules are not considered sufficiently interesting, then a different algorithm (or the same algorithm with different parameters and restrictions) can be applied to discover a more interesting group of rules.
4. **Improving the ASWE.** Based on the discovered relationships, the teacher can perform the modifications that he considers most suitable for improving the ASWE. For instance, he can modify the course's structure (joining concepts, changing the level of concepts or the chapter in which they are presented, etc.) and modify the course content (eliminating or improving bad questions, bad content pages, etc.). To do this, he can again use an authoring tool.

The described process can be repeated again and again as soon as enough information from new students has been collected. This iterative process allows the teacher to progressively improve the ASWE as more students use it.

Our aim is to discover important dependencies in the usage data that is recorded during students' learning sessions with ASWEs (e.g. the Linux course described in Subsection 4.1). Specifically we aim at finding relationships that can be expressed as IF-THEN rules. The IF-THEN rule is one of the most popular

forms of knowledge representation, due to its simplicity, comprehensibility and expressive power (Klösgen and Zytkow, 2002). Depending on the represented knowledge there are different types of rules. In the area of knowledge discovery in databases, the most studied ones are:

Association rules The purpose of association rules (Agrawal et al., 1993) is to look for relationships among attributes in databases (the respective attributes become part of the antecedents and consequents of the rules). Association rules are typically used in e-commerce to model the clients' preferences and purchases. Such rules have the format: IF "user acquires product A" THEN "user also acquires product B", with values of *support* and *confidence* (Agrawal et al., 1993) higher than a user-specified minimum threshold. In the more general form of these rules, the rule antecedent and consequent can comprise 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.

Classification rules The function of classification rules (Quilan, 1987) is to obtain the necessary knowledge to create a classification system (similar to a classification tree). The antecedent of a rule contains requirements (in the form of conditions), which match those object that belong to the class that is identified in the consequent of the rule. From a syntactic point of view, the main difference to association rules is that classification rules have a single condition in the consequent which is the class identifier name.

Prediction rules The objective of prediction rules (Noda et al., 1999) is to predict a target attribute based on the values of other attributes. Like classification rules, they only have a single condition in the consequent, which can however be any attribute. 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.

We are going to discover prediction rules through a dependence modeling task. This data mining task consists of the prediction of relations between attributes, which may or may not be specified by the user (Freitas, 2002). Dependence modeling can be viewed as a generalization of classification rule discovery, or a specialization of association rule discovery. However, dependence modeling involves a wider notion of dependence than is the case for classification, 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 target attribute or class can only occur in the consequent and the prediction attributes only in the antecedent. Although the association task is symmetric with the attributes, as 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.

In order to demonstrate the value of our methodological proposal we have to prove that there exist algorithms for prediction rule discovery that can yield useful information for the improvement of an ASWE. We will show this by using data from an adaptive course on Linux that we developed. In the following Section, we will describe our ASWE, the usage data that we collect from it, and the data preparation process to which we subject this data.

4. Collecting and Preparing Usage Information from our ASWE

4.1. A LINUX COURSE BASED ON AHA!

Web servers normally log the requested files only, the times when the requests were made, and the IP number of the requester. Even though this information is commonly the basis of web mining, we need more information about each student's interaction (Heift and Nicholson, 2000): scores obtained in activities, times for reading pages, knowledge level achieved in the assessment of students' familiarity with concepts and chapters, etc.

We developed an adaptive web-based course of Linux using AHA! version 1.0 (Romero et al., 2002) to obtain the usage information we needed. AHA! is a general architecture for developing adaptive hypermedia applications. We selected AHA! to build and enhance 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 text fragments, and disabling 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, in the following manner (Romero et al., 2002):

Domain Model. Like many other contemporary ITS systems, our course material is stratified into several difficulty levels. A course consists of several chapters that are organized in lessons. Each lesson includes several concepts, and each concept is assigned to a difficulty level (namely HIGH, MEDIUM or LOW).

User Model. User models in AHA! consist of a set of concepts with attributes. We changed students' knowledge attribute for concepts and chapters: while it ranges from 0 to 100 in AHA!, we only introduced the discrete values 0 (NOT YET READ), 10 (BEGINNER), 50 (NORMAL) and 100 (EXPERT).

Adaptation Engine. We perform the adaptation from a chapter view point (see Figure 2). Before they begin with a new chapter, students have to take an initial adaptive test to determine their initial knowledge level. Then the system presents only those concepts to them that have a suitable difficulty level. Students have to read the instructional pages and perform the assessment activities for each concept. Eventually they have to take a final test to evaluate

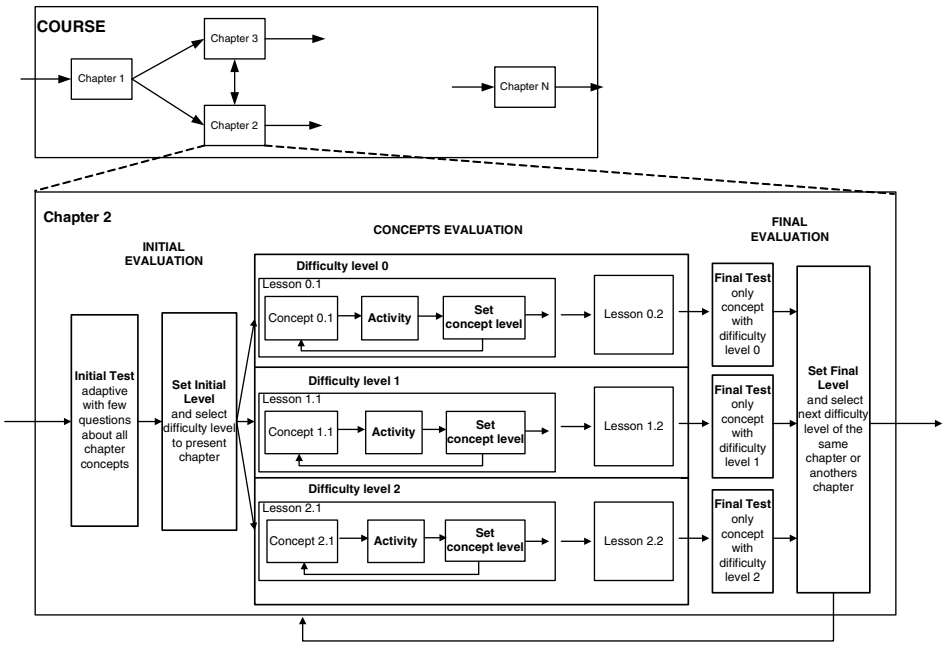


Figure 2. Modified AHA! adaptation engine.

their knowledge about the chapter. Students may thereafter repeat the chapter at the same level (if they obtained the BEGINNER level), go on to a higher level for the same chapter, or pass to another chapter (if they obtained a NORMAL or EXPERT level). This process repeated for every chapter.

Based on this modified AHA! version, we developed a course about the operating system Linux. Our changes to the original AHA! system enable us to adapt the hypermedia presentation to the *knowledge level* of each student. In Figure 3, the same chapter from the Linux course is shown at three different difficulty levels: BEGINNER (left), NORMAL (middle) and EXPERT (right). Each version starts with a different concept explanation that is suited to the respective knowledge level. This presumed knowledge level is also expressed by the background color of the pages and the text label: Nivel 0 (BEGINNER), Nivel 1 (NORMAL) and Nivel 2 (EXPERT). The Linux course has been taken by 50 students in Computer Science Engineering at the University of Córdoba.

The original AHA! (1.0) stores information about each students' navigation and knowledge in two different files. We added another log file to store the scores of activities and test questions. For each student, the specific contents of these three files are:

Log is a text file that contains information about the name and the time (in seconds) of each visited page. It is similar to the traditional web log file.

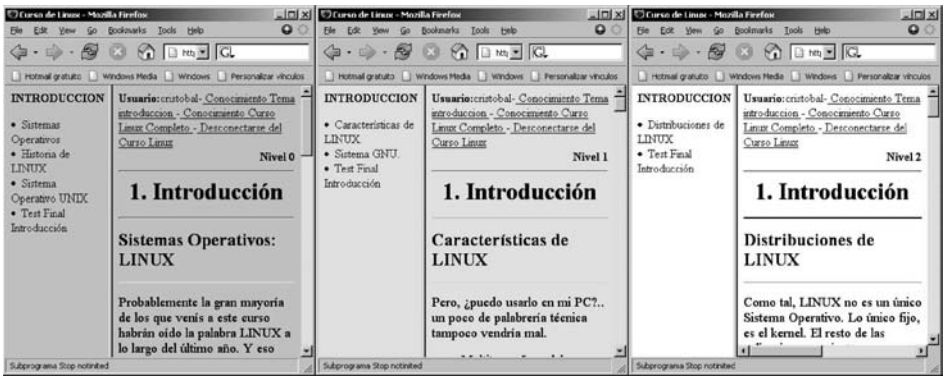


Figure 3. Three different levels of the Introduction Chapter of the Linux course.

Model is an XML file that records the knowledge levels that students have attained for each concept and chapter, in a discrete numerical form (0, 10, 50 or 100).

Test is a text file with information about students' successes or failures in the test and activities questions (YES or NO value).

4.2. DATA PREPARATION

All information in the three log files must be transformed and stored in (relational database) before data mining algorithms can be applied, to facilitate the algorithms and increase their speed when searching for rules. The following preprocessing tasks are being carried out during this process (Freitas, 2002).

4.2.1. *Attribute Selection*

The main goal of attribute selection is to choose a subset of relevant attributes from all available attributes of the data being mined. We selected these attributes manually, namely *user name* (student name), *course* (name of the course), name (name of the page, concept or chapter), *type* (instructional 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).

4.2.2. *Data Cleaning*

This activity consists in looking for erroneous or irrelevant data and discarding it. We discovered several types of errors: long times (longer than 10 minutes) and incomplete data (incompletely visited chapters, and unfinished tests and activities). We also discovered several types of irrelevant data: container pages (frame pages),

index pages (table of contents pages), help pages and log out pages. We discarded all this information.

4.2.3. *Transforming Continuous Attributes*

This activity consists in 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 only transformed the attribute time and assigned three values or labels to it (HIGH, MEDIUM and LOW), using the equal-width method.

4.2.4. *Data Integration*

The goal of data integration is to group together all the data from different sources. In our case, we gathered all preprocessed data from the three log files of each student in a MySQL relational database (Dubois, 2003). We used MySQL because it is portable, fast and free. In Figure 4 we show the relational scheme of the students' usage information database that we used.

5. Evolutionary Algorithms for Rule Discovery

Rule discovery is a classical problem, which has been approached using different methods such as the construction of decision trees, inductive learning, instance-base learning, and more recently neural nets and evolutionary algorithms (Witten and Frank, 2000). Of these, decision tree construction is currently most frequently used in data mining. These algorithms are very fast and surprisingly effective in finding precise classifiers. As they use greedy heuristics to divide data, they may however fail to find some multi-variable relationships. But there exists alternative algorithms that aim at conducting more meticulous searches. One such example are evolutionary algorithms (EAs), which can implicitly backtrack when searching a

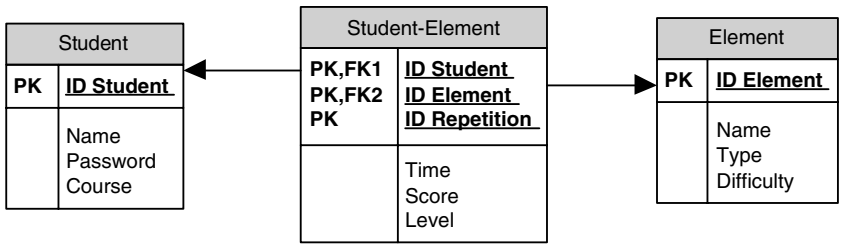


Figure 4. Database relational scheme.

rule space. This ability allows them to find complex interactions among attributes that other types of algorithms are not able to find.

In the context of rule discovery using EAs (Freitas, 1997), an individual represents a rule or a group of candidate rules. The fitness function corresponds to some evaluation measure for the rule quality. A selection procedure uses the fitness value to choose the best rules. Genetic operators transform the candidate rules into new rules. The main advantages in adopting EAs are the ability to work in a search space thoroughly, and the ability to allow arbitrary fitness functions in the search (Dhar et al., 2000). Main disadvantages are their low speed and the randomness of their 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 rule induction, 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 taken 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.

Prediction rule discovery with evolutionary algorithms can be carried out with two different approaches: restricted or unrestricted. In the restricted approach (Freitas, 2002), the problem is treated as classification rule discovery in which users have to specify the attribute(s) that are to be predicted. That is, individuals only represent the rule antecedent conditions. The objective is to discover the best conditions that predict the previously set attributes. In the unrestricted approach (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. For our purposes we are going to use the unrestricted approach, but users can still specify some filters to find certain types of rules (with or without a specific type of condition in the rule antecedent or consequent, a maximum number of conditions in the rule, etc.)

5.1. GRAMMAR-BASED GENETIC PROGRAMMING FOR DISCOVERING PREDICTION RULES

Within Evolutionary Algorithms, several subtypes can be distinguished, such as Evolutionary Programming, Evolutionary Strategies, Genetic Algorithms (GA) and Genetic Programming. The latter two approaches are the most common for rule discovery. GAs for rule discovery can be further divided into two main approaches (Freitas, 2001), based on how rules are encoded in the population of individuals. In the Michigan approach, each individual encodes a single prediction rule, and in the Pittsburgh approach a set of prediction rules. The predominant 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 encoding

of individuals simpler and syntactically shorter. In 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 similar to a version of Genetic Algorithms that uses trees to represent individuals. Although GA is a more widely used, GP can be considered as the 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. However, there is a problem when rules are encoded into a GP individual, due to the closure property of GP (Freitas, 2001). This property requires that the output of a node in a tree can be used as the input to any parent node in the tree (that is, both nodes have to belong to the same type). There are different approaches of GP that cope with the requirement of closure, such as Strongly Typed (or "Constrained-Syntax") Genetic Programming and Grammar-Based Genetic Programming (GBGP).

In GBGP, individuals are represented as a derivation tree of a user-defined grammar to specify the problem solution space (Whigham, 1995). GBGP systems use grammars to impose syntactical constrains on programs. The use of grammars also helps one to overcome the closure requirement in canonical genetic programming, which cannot always be readily met. The grammar can be used to enforce elaborate semantic restrictions based on the domain knowledge provided by a domain expert. We chose GBGP due to its expressivity and its ability to interact with users, who can select different types of desired rules by restricting only the grammar. GBGP demonstrated high performance on a number of problems and has been considered as one of the most promising areas in genetic programming (Nordin et al., 1998). In the following, we will describe several pieces of work on the use of Genetic Programming for rule discovery.

One of the first systems that use GP for knowledge discovery is MASSON (Ryu and Eick, 1996). It focuses on discovering the common characteristics of a set of objects in 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 evaluates them to see whether queries compute the same set of objects that are given by the user.

In related work, (Ratle and Sebag, 2000) use genetic programming for the discovery of empirical laws. They propose a way for enforcing dimensional constraints through formal grammars, to restrict the GP search space to dimensionally admissible laws. They use grammar rules to incorporate dimensionality constraints

into GP, and an initialization procedure based on dynamic pruning of the grammar to generate only feasible trees of a prescribed derivation depth.

The LOGic grammar-based GENetic PROgramming system (LOGENPRO, (Wong and Leung, 2000)) uses Inductive Logic Programming (ILP) and genetic programming to impose syntactic and semantic restrictions. The authors describe a framework, called GCP (Generic Genetic Programming), which integrates genetic programming with a formalism of logic grammars. This formalism is powerful enough to represent context-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.

In other research (Freitas, 1997) uses a genetic programming framework for classification and generalized rule induction. He emphasizes the integration of a GP algorithm with relational database systems, which leads to decrease data redundancy, and to improved scalability, portability and control of data privacy.

Finally, (Bojarczuk et al., 2001) use genetic programming to discover classification rules for diagnosing certain pathologies in medical databases. They employ constrained-syntax GP, in which some restrictions must be satisfied to obtain a valid rule. They use several medical databases (e.g. on chest pain, dermatology and breast cancer) for discovering high-level comprehensible classification rules.

As we can see, most of the work using Genetic Programming for rule discovery is focused on classification rules. In this specific approach, the rule consequent is not a typical condition (attribute-value pair) but a single name (namely the name of the class). Consequently, many approaches only encode the antecedent of the rule in the individuals. Our approach is more general in that we want to discover prediction rules that are more general than classification rules (see Section 3). The main difference in our approach is that we encode both the antecedent and the consequent of the rules in the individual. We are going to describe our approach to GBGP for discovering prediction rules in the following.

5.2. ENCODING OF INDIVIDUALS

In the encoding of individuals 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. In our case the genotype is a syntax tree of instructions, which we implemented as a list of integers (Ventura et al., 2002).

Phenotype. The phenotype is the meaning of the genetic material of 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).

Table I. 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 II. Functions and arguments

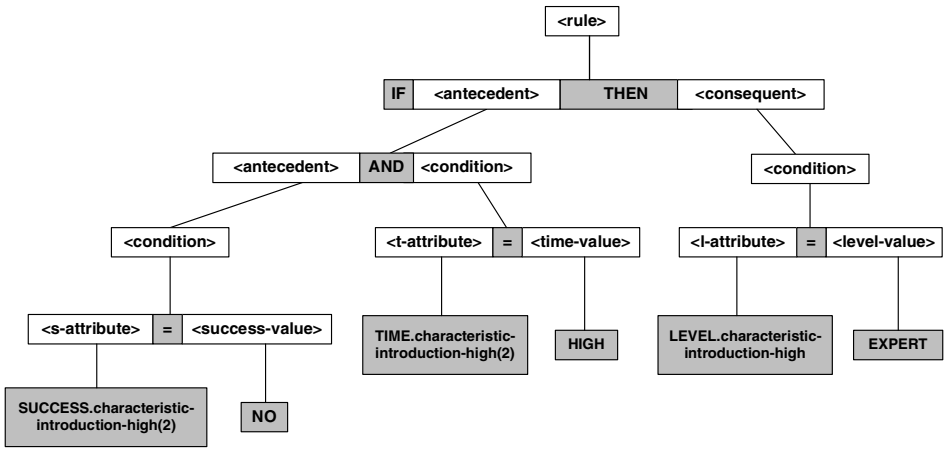
Functions	Input arguments	Output arguments
IF THEN	boolean	boolean
AND	categorical	boolean
=	categorical	categorical

Table I shows the grammar that we use to generate the individuals that represent prediction rules. 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 knowledge level), with one possible value for the attribute. We do not show all terminal symbols of valid attribute names since there are too many (namely all names of concepts, chapters, and instructional pages, activity pages and test pages). All attribute values are categorical or nominal. Table II shows the functions or non-terminal symbols of our grammar.

Figure 5 shows an example rule generated by our grammar. The meaning of this rule is that students, when rated as EXPERT in the concept CHARACTERISTIC in the INTRODUCTION chapter at the level HIGH, fail in question number two of the activity of this concept, and they also need a HIGH time to answer that question. This rule thus shows that something is wrong with this question. The teacher should review it and decide whether to change the explanation or the answers of the question.

5.3. EVOLUTIONARY ALGORITHM

The evolutionary algorithm that we implemented to discover prediction rules is a generational GP with an external elite file. This means that we use two different groups of individuals: one represents the current population in each generation, and the other represents the elite population with the best individuals that will be



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

Figure 5. Derivation tree of example rule.

Table III. 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.
    Apply genetic operator over selected parents.
    Evaluate obtained children with a single or multi-objective approach.
    Update individuals in P and add the best new individuals to E.
    Increase current-generation by one.
  End While
End
```

finally returned to the user as the set of discovered rules. Table III shows the evolutionary algorithm.

The first step is to create the initial population P (whose size is fixed) from the initial data chosen by the user (see Section 5.4). We must also create an external file E (with variable size) that is initially empty, to store the best individuals of the current population in each generation step. From the current population P and elite file E we then select the parents that are to be reproduced (see Section 5.7). The children are generated by applying genetic operators (see Section 5.5). Later they are evaluated using a multi-objective or single-objective approach (see Section 5.6). Thereafter the elite population is updated by adding the best new individuals of the current population P. This process is repeated until a maximum number of generations has been reached. In the following, we will describe these steps of our GBGP algorithm in more detail.

5.4. INITIALIZATION

Initialization consists in generating a group of initial rules. The teacher must select the data that will be used to construct them. He can choose all available values, a range of values (those with a relative frequency greater than a specific threshold), frequent values, or infrequent values. These initial elements are the only ones used to generate initial rules. There are two reasons for allowing the use of different initialization data. The first is to compare the performance of the algorithm with different types and amounts of data: a large amount (namely all data), a small amount (the most frequent data), and an average amount (a range of data). The second reason is to prove the merits of using a representative data set instead of all data. We propose to use range data rather than all data, or only very frequent or infrequent data which apply to almost every or hardly any student.

Then we compose the initial rules from these selected data by deciding randomly which elements will be put into the antecedents and the consequents of the rules. The user can specify a maximum size for all rules. The size of the rule varies dynamically depending on the number of elements in the antecedent. The last element always represents the consequent. After creating the initial population, we have to verify that the rules are always correct. Although all rules generated by our grammar (see Table I) are syntactically correct, some of them may be semantically incorrect (e.g., rules with the same condition in the antecedent and consequent, or rules with a repeated condition in the antecedent). This problem is caused by the fact that we use a context-free grammar. To solve it, we use a specific repair operator (see Section 5.5.3).

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

5.5. GENETIC OPERATORS

We use three genetic operators, namely selective crossing, selective mutation (Whigham, 1995) and a specific repair operator.

5.5.1. *Selective Crossover*

Selective crossing is comparable to crossing trees in genetic programming, where two subtrees of each parent tree are mixed to form two new child trees. But in selective crossover, 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 be exchanged have to be semantically compatible. Selective crossover can perform five different exchange operations on rules, namely on the antecedents, consequents, conditions, condition attributes and condition values. We can vary the probability of each operation when we configure the parameters of the algorithm.

5.5.2. *Selective Mutation*

Selective mutation is also comparable to mutation trees in genetic programming, where a subtree 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 chosen as the root node in the mutation.

5.5.3. *Repair*

We use a specific repair operator to fix incorrect individuals. An individual can be syntactically correct (generated by the grammar), but semantically incorrect (like e.g. a rule with the same condition in the antecedent and consequent, or with duplicate conditions). To solve this problem, we use the "Reparation" operator which performs a simple mutation on the condition until the problem disappears.

5.6. EVALUATION

Evaluation consists in computing the fitness function, i. e. the quality evaluation function of the current rules. There exist numerous rule evaluation measures (Lavrac et al., 1999; Tan et al., 2002; Yao and Zhong, 1999) from statistics, machine learning and data mining (see Appendix A). These measures try to evaluate different features of the rule (precision, interest, reliability, comprehension, simplicity, etc.), and there is no single measure that is optimal in all application domains. It therefore seems best to use several measures and to optimize not only a single function but several functions simultaneously, similar to multi-objective optimization (Fonseca and Fleming, 1993). There are different approaches to solve this problem of multi-objective optimization with evolutionary algorithms: one approach is to use aggregation functions, another is to use the concept of Pareto Front.

5.6.1. *Aggregation Function*

In the case of aggregation functions, the evaluation function is a linear combination of different measures that need to be optimized (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 for rule discovery in the literature, including the following ones:

– The aggregation function of (Araujo et al., 1999) consists of two components (see 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.

– The aggregation function of (Liu and Kwok, 2000) consists of three components (see Equation 2): the first is the Laplace function (Bayardo and Agrawal, 1999) to measure the consistency of the rule, the second represents its completeness, and the third its comprehensibility (Liu and Kwok, 2000).

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

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

– The aggregation function of (Freitas, 2002) consists of two components (see Equation 3): the first represents the accuracy and the second the comprehensibility of the rule (Liu and Kwok, 2000).

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

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

5.6.2. *Pareto Front*

The algorithms based on the concept of Pareto Front (Fonseca and Fleming, 1993) use a vector of objectives to optimize within individuals. The purpose is to make populations converge towards the group of best solutions (which are dubbed Pareto Front). In this way, the final solution is the best in terms of all objectives combined and not in any specific one. There are different algorithms within this approach, including:

– 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 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 (Non-dominated Sorting Genetic Algorithm) (Srinivas and Deb, 1994) is based on several steps of the 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 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 be used. In our case, we have to choose what the rule quality evaluation measures are which we want to optimize. The knowledge discovered by a data mining algorithm should satisfy three main aspects (Freitas, 2002): it should be accurate (certainty), interesting (novel, surprising, useful) and comprehensible (simple).

Accuracy. The concept of rule accuracy (Lavrac et al., 1999) that we 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. So we measured the accuracy of the discovered rules using the whole data set, as is done in association rule mining, and did not use different test and training sets as done in classification.

Interestingness. Rule interestingness (Piatesky-Shapiro and Metheus, 1994) can be measured using two types of measures: subjective (user-driven) (Silberschatz and Tuzhilin, 1995) and objective (data-driven) (Tan et al., 2002). We use constraints of the user regarding the rules he wants to discover as well as an objective measure of rule interestingness.

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

We use these three criteria to measure the quality of discovered rules. We employ a three-dimensional vector where each dimension measures one of these aspects. The specific measures we use are:

- **Certainty Factor**. The Certainty Factor (Shortliffe and Buchanan, 1975) is a measure of the rule precision. 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

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

  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 outperform the classic Piatetsky-Shapiro measure of interest (Silverstein et al., 1998). The interestingness of a rule $A \rightarrow C$, is

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

  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 of rule compressibility so that the shorter the rule, the more comprehensible it is. The simplicity of a rule $A \rightarrow C$, is

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

  where AntecedentSize is the number of conditions in the antecedent and MaximumSize is the maximum number of conditions in the antecedent. We have normalized the measure to take on values in the range of zero to one.

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

5.7. SELECTION

Selection consists in the election of two 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 (linear ranking) that first ranks the population according to its evaluation value (fitness function value) and then every rule receives its final fit from its ranking (Michalski, 1998). Hence, the worst rule (the rule with lowest fitness value) will receive fitness 1, the second worst fitness 2, etc., and the best will obtain fitness N, where N is the number of rules in the population. Parents are selected according to their fitness. With this method, all 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 representing the number of different conditions in the antecedent and consequent of each rule. The individuals with a higher value in this metric are structurally the most different and will be more probably elected. We also assure that there are no repeated individuals in the population, in order to avoid the problem of premature convergence.

6. Implementation and Experimental Results

In this section, we describe the different knowledge discovery algorithms that we implemented, and the specific software for applying them to improve ASWEs. We will also describe different tests that were performed and compare the results from different algorithms.

6.1. IMPLEMENTATION OF EPRULES

To facilitate the complete process of prediction rule discovery, we developed a special tool named EPRules (Education Prediction Rules, see Figure 6). It is intended directly for the teacher or the course developer. Non-experts in data-mining can use it much more easily than other generic tools such as DBMiner (Klösgen and Zytkow, 2002) and Weka (Witten and Frank, 2000). On the other hand, it is more powerful for discovering knowledge in ASWEs than general-purpose tools since it is a special-purpose tool and uses specific domain knowledge.

The main components of the EPRules tool interface (Figure 7) are:

Data input. This component allows one to open an existing database with course usage data, or to create a new database and to add and preprocess new student usage information. The parameters for transforming the time-variable can also be selected and configured (see Section 4.2.3). When duplicate data is encountered (e.g., pages that were visited or activities that were performed more than once), EPRules can only use one instance. By default, it will use the last record but can be configured to use the first.

Data viewer. This component allows us to visualize students' usage data and to compute basic statistics (maximum, minimum, average, etc.). It can selected to visualize the data of all or a specific student, and to restrict the display further to a particular chapter of the course, a specific concept of a chapter, a specific

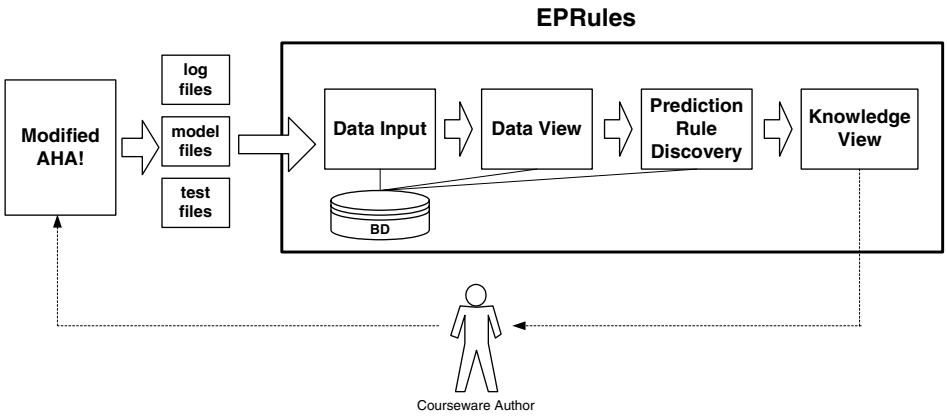


Figure 6. EPRules interface components.

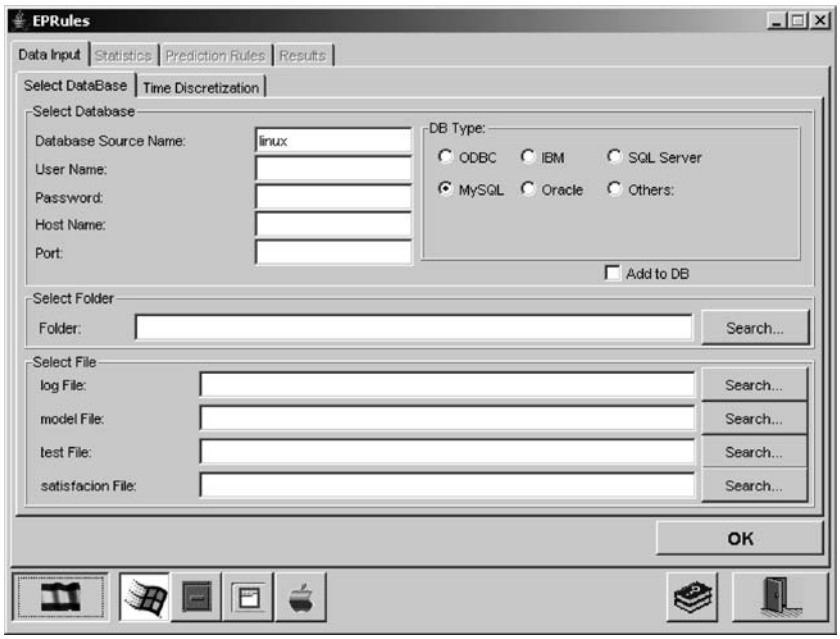


Figure 7. The data input window in the EPRules tool.

difficulty level of a chapter (high, normal, low), or a particular type of information (time, level or score).

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 us to:

1. select one of the available rule discovery algorithms: ID3 (a decision trees construction algorithm, (Quilan, 1987)), Apriori (an association rule mining algorithm, (Agrawal et al., 1993), Prism (covering algorithms, (Cendrowska, 1987)) or GBGP;
2. set the specific execution parameters of each algorithm;
3. select the subjective restrictions that rules should match (see Figure 8), namely just one particular chapter or concept, a single student, and a specific knowledge level, score or time; and
4. choose the objective evaluation function, so that the discovered rules are really useful to the teacher.

Knowledge viewer. This component allows us to visualize the discovered prediction rules in a window. On the left side it shows the antecedent and consequent conditions of the rules, and on the right side the value of all rule evaluation measures (confidence, support, interest, gini, laplace, etc., see Appendix A). The window with the knowledge view appears automatically after the rule discovery algorithm finished its job. Rules are listed in the order of their discovery, but can also be ordered differently based on the value of any measure or condition, or can be manually rearranged by clicking at one of the columns.

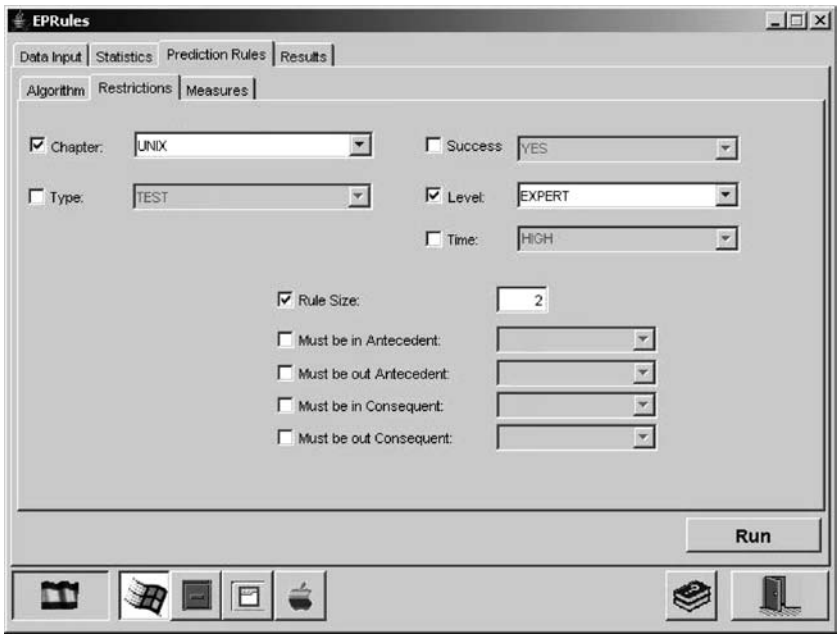


Figure 8. The restrictions window in the EPRules tool.

EPRules includes three classic knowledge discovery algorithms written in Java, namely ID3, Prism and Apriori. We chose to implement these algorithms since they are among the most representative methods for rule discovery (Witten and Frank, 2000), and since other authors have previously used them in comparison tests with new proposed algorithms (Freitas, 2002; Hipp et al., 2000). We had to adapt these algorithms to the specific discovery of prediction rules though. The only change that we applied to these algorithms was to ensure a common rule format. The conversion from an association rule algorithm to a prediction rule algorithm is trivial, since it is only necessary to force rule consequents to have a single condition. When converting classification rule algorithms and covering algorithms into prediction rule algorithms, the consequent or class of the rule can be any condition (any attribute value pair). Moreover, if we want to extract rules with N different attributes, we have to run the algorithm N times, using a different attribute as the condition of the consequent in each case.

The GBGP algorithms (with 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 encodes syntactic trees as vectors of ordered integers that represent the pre-order traversal of the tree. In order to evaluate the individuals it is necessary to transform the list of integers into several SQL queries (Sarawagi et al., 1998), to determine the values of the evaluation measures used in each case.

Table IV. 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
$< antecedent >, < consequent >$ roulette value	1
$< condition >$ roulette value	4
$< attribute >, < value >$ roulette value	2
Mutation operator	Selective mutation
Probability of success	0.2
$< antecedent >, < consequent >$ roulette value	1
$< condition >$ roulette value	1
$< attribute >, < value >$ roulette value	1
Stop stage	
Maximum number of generations	50

6.2. DESCRIPTION OF THE EXPERIMENTS

We have used the usage data of 50 users of our adaptive Linux course (see Section 4.1). We carried out two types of tests to compare the implemented knowledge discovery algorithms: one for the number of discovered rules and the computational costs and the other for the quality of these rules. We also performed these tests on populations of three different sizes. More precisely we used all available data, frequent data only (with a relative frequency higher than 0.5), and range data only (with a relative frequency between 0.2 and 0.9). In the case of evolutionary algorithms we ran ten cycles, using the parameters shown in Table IV.

We want to compare classic algorithms with evolutionary algorithms under similar conditions without giving either of them an undue advantage. We therefore used the same type of rule (prediction rule), the same initial data elements and the same maximum number of elements in the antecedent of the rule (namely three). We also used typical parameter values for each algorithm that needs them (minimal support and confidence, number of generations, etc.).

6.3. PERFORMANCE OF THE ALGORITHMS

We counted the number of rules that the implemented algorithms discovered under EPRules, and determined the percentage of accurate, interesting and comprehensible rules. We then compared classic algorithms (ID3, Prism and Apriori) with the average result from the mentioned earlier evolutionary algorithms (Liu, Freitas,

Table V. Total number of rules and number of accurate, interesting and comprehensible rules discovered by each algorithm using all data

	Total	Accurate	Interesting	Comprehensible
ID3	474	218	10	42
Prism	657	473	16	26
Apriori	5960	4529	244	774
Liu	317	200	43	72
Freitas	284	173	59	68
Araujo	350	210	141	17
MOGA	98	82	56	24
NSGA	75	69	55	19

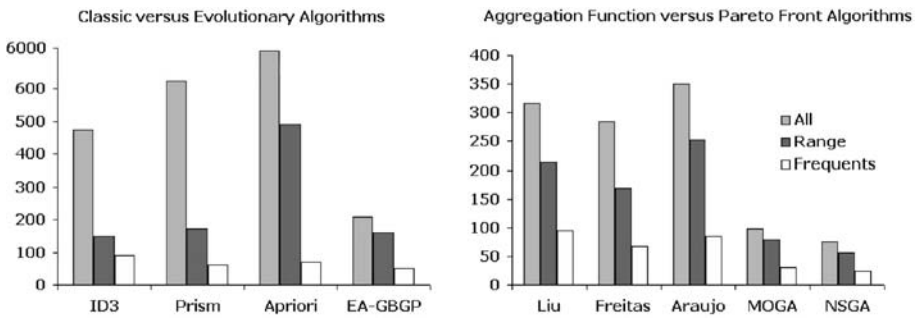


Figure 9. Total number of discovered rules.

Araujo, MOGA and NSGA), and also these evolutionary algorithms individually with each other.

Number of Discovered Rules. Figure 9 shows the total number of discovered rules for classic algorithms, evolutionary algorithms, and the average of different versions of GBGP (note the scale difference between the left and right diagrams). We see that classic algorithms discover a huge number of rules, especially when all data is used and also with the Apriori algorithm. This effect is attenuated by a decrease in the population size (cf. range data and frequent data). On the right-hand side, we see that the algorithm that discovers the lowest number of rules is NSGA, followed by MOGA. Table V shows the total number of rules and the number of accurate (certainty > 0.7), interesting (interesting > 0.5) and comprehensible (simplicity > 0.5) rules obtained by using all data. We see that Apriori algorithm always generates a bigger number of accurate, interesting and comprehensible rules.

Total Execution Time. Figure 10 shows that evolutionary algorithms are much faster than classic algorithms when all data is used. But the fastest algorithm for frequent data is Apriori, which is not surprising. On the right-hand side we see no significant differences between the implemented evolutionary algorithms (note again the different scales).

Percentage of Accurate Rules. Figure 11 shows the percentage of accurate rules (with a certainty factor greater than 0.7). We see that Apriori discovers a very high percentage of accurate rules, and so do the algorithms based on Pareto Front (MOGA and NSGA).

Percentage of Interesting Rules. Figure 12 shows the percentage of interesting rules (with an interestingness greater than 0.5). We see that only some of evolutionary algorithms discover a higher percentage of interesting rules than classic algorithms. These best results are obtained with algorithms based on Pareto Front and Araujo's Aggregation Function (Araujo's algorithm contains a component related to interest).

Percentage of Comprehensible Rules. Figure 13 shows the percentage of comprehensible rules (with a simplicity value greater than 0.5). We can see that some evolutionary algorithms discover a higher percentage of comprehensible rules, especially algorithms based on Pareto Front and Liu's and Freitas' Aggregation Function (Liu's and Freitas' algorithms contain a component related to simplicity).

We are now going to summarize the main results from the previous comparisons.

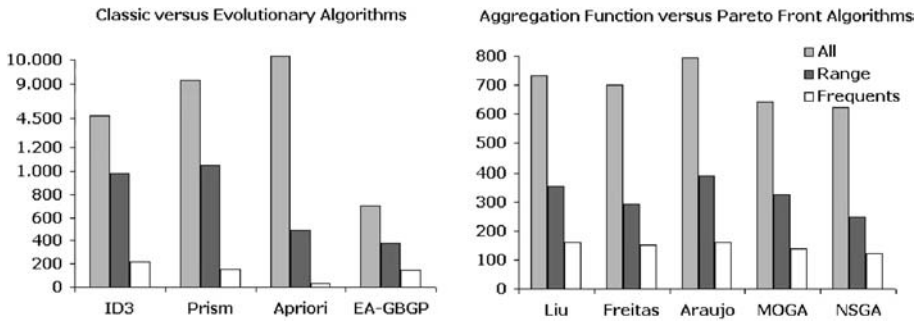


Figure 10. Total execution time (in seconds).

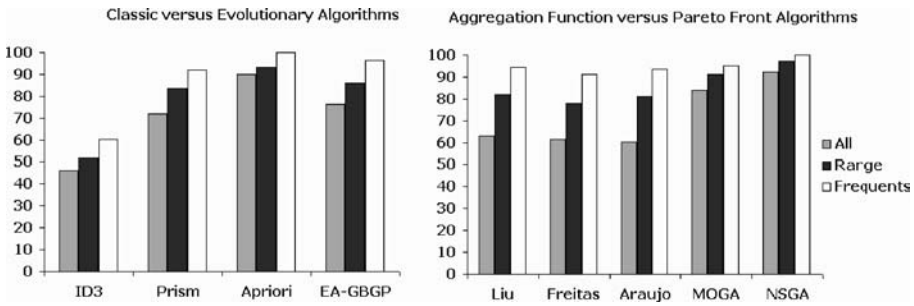


Figure 11. Percentage of accurate rules.

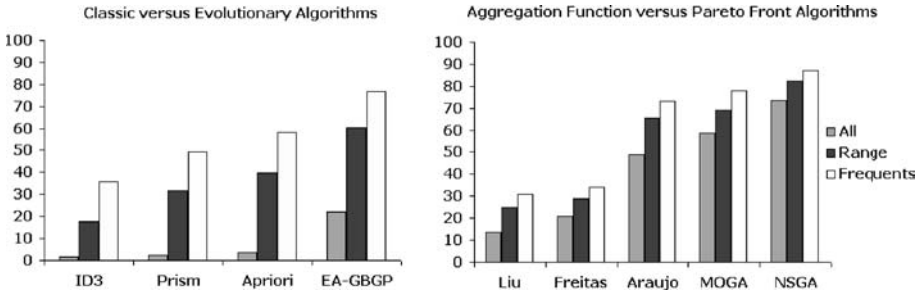


Figure 12. Percentage of interesting rules.

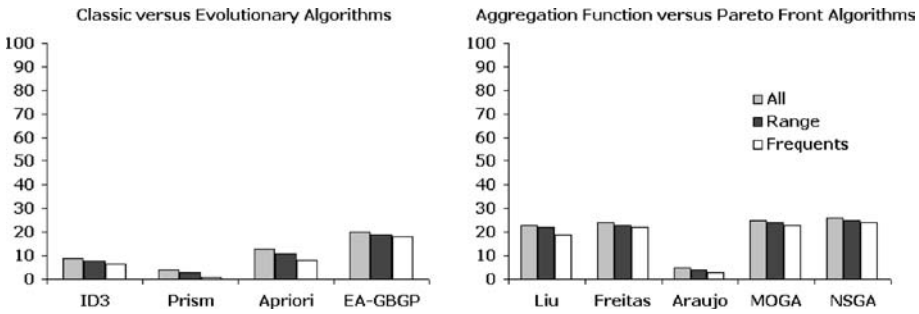


Figure 13. Percentage of comprehensible rules.

In general, classic algorithms (mainly Apriori) discover a bigger number of very accurate rules and are very fast when applied to frequent data. However, the percentage of short rules with high interestingness is low for them (although the total number of short and interestingness rules or the absolute values of some them can be higher than EA-GBGP algorithms). When applied to lots of data (all available data), they also require long execution times and generate so many rules that it becomes hard to use these directly and to identify the most useful ones. But the usefulness of the rules is a subjective measures that depend on the users who examine the rules (Silberschatz and Tuzhilin, 1995). A well-known criticism of many rule discovery algorithms in data mining is that they generate too many rules (Padmanabhan and Tuzhilin, 2000). In general, the fewer rules are in a rule set, the more comprehensible it is (Freitas, 2002). So, it will be difficult for the user to comprehend so many rules and to identify the useful ones (Liu and Hsu, 1996). Additional postprocessing therefore becomes necessary to filter or prune it (Jaroszewicz and Simovici, 2002; Liu and Hsu, 1996) in order to obtain smaller sets of useful rules.

On the other hand, multi-objective EA-GBGP algorithms can directly produce a smaller number of rules than classic algorithms, and in in much less time when all data are used. Also, the percentage of comprehensible and interesting rules can be significantly higher (i.e., discovered rules can be used directly with no need for additional filtering). Among the approaches based on aggregation functions, Liu and Freitas focus on optimizing the accuracy and comprehensibility of the discovered rules, and Araujo on their interestingness. However, algorithms based on the concept of Pareto Front

(MOGA and NSGA) can simultaneously optimize the three objectives, discovering the biggest percentage of accurate, comprehensible and interesting rules.

A final comment concerns the scalability of the algorithms in terms of the speed of the rule discovery. Classic algorithms are fast when the size of data is small (frequent data), but they do not scale well. They become extremely slow when the size of data grows (range data and all available data). In contrast, the performance of evolutionary algorithms is less dependent on the size of the data, and their speed for generating rules is less variable than that of classic algorithms.

6.4. INTERPRETATION OF DISCOVERED RULES

The course author has a crucial role in this form of rule mining since he can guide the search by imposing subjective restrictions (see Figure 8), using his own knowledge and experience in education. The author can decide, e.g., to use all available data, frequent data or range data only, to use data about one specific chapter or rather about the whole course, about all students or only students with a final knowledge level of EXPERT or BEGINNER, and to only use data about times, levels and successes to construct rule antecedents and consequents.

It is important to mention that the comprehensibility and interestingness of rules are subjective concepts that are 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 for the mining task.
2. Rule constraints: the teacher can select specific constraints on the rules to be mined.
3. Interestingness constraints: the teacher can specify the values or ranges of a measure that are interesting for him.

As mentioned before, our objective is to show a group of useful rules to the teacher, so that he can decide on course improvement. Semantically our discovered rules express the following relationships:

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

Level, Time and Success are thereby expressions referring to users' attained knowledge state (BEGINNER, NORMAL, EXPERT), the reading time for pages (HIGH, MEDIUM, LOW), and to information on students' successes and failures in the test and activities questions (YES, NO). More details can be found in Table I. Based on the discovered rules the teacher can decide which of the expressed relationships are desirable or undesirable, and what can be done to strengthen or weaken them (namely changing or modifying the contents, structure and adaptation of the course).

The relationships that are expressed in discovered rules can refer to chapters, concepts, or scenarios of concepts (namely instructional and activity pages relating to concepts). We correspondingly distinguish three types of rules:

Rules about chapters describe relationships between different chapters of the same course. They refer to knowledge levels obtained in an initial or final test, and have the following pattern:

IF ChapterLevel **AND** ... **THEN** ChapterLevel
Where ChapterLevel are conditions about tests levels.

Using this information, the teacher can decide, e.g., to change the sequence of chapters in the course, to merge them into one chapter if he wants to increase a relationship, or to move them further apart in the course if he wants to decrease it.

Rules about concepts show relationships between different concepts of the same or different chapters. They refer to knowledge levels obtained in activities and have the following pattern:

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

Using this information, the teacher can opt to put activities into the same chapter if they are in different chapters, to put them at the same difficulty level if they have different difficulty levels, to put next to each other in the chapter, or rather to put them further apart within the same chapter or in other chapters of the course.

Rules about scenarios of concepts show relationships between scenarios of type instructional and/or activity. They refer to times, successes and levels obtained in instructional pages and evaluation activities for concepts. Their pattern is:

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

Using this information, the teacher opt to, e.g., delete pages because they are duplicate, badly phrased or incorrect; 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.

6.5. DISCOVERED RULES IN OUR LINUX COURSE

In this section, we are going to describe the meaning and the possible use of several rules regarding our Linux Course that we discovered 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 two different concepts have been simultaneously very high (EXPERT). This may indicate 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 search both concepts in the educational material and try to find out why they are related. He should also decide if merging both concepts into a single concept, putting both concepts into the same chapter, making their level of difficulty equal, correcting the rules that assign levels, or any other modification would be helpful. In this specific example we decided that both concepts should have the same level of difficulty (namely HIGH).

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 describes the fact that whenever question 0 of the Administration chapter takes long to read and answer, then it is often answered wrongly. This relationship may indicate that the question is not well articulated or has some kind of error. When the teacher discovers this type of relationship, he should correct it by modifying the wording of the question or by replacing it with some other question. In this concrete example we found the wording of the question corresponding to the ADMINISTRATION concept to be confusing, and we had to replace it with a different (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 that students who are regarded as EXPERTS with respect to the concept EMULATORS in the PROGRAMS chapter fail question 1 in the evaluation activity for this concept. Such a finding may again indicate that this question is not well articulated, not be well enunciated. In this specific example we found that question 1 was confusingly worded and rewrote it to solve the problem.

6.6. USE OF EPRULES WITH OTHER ASWES

Although we only used EPRules in combination with AHA! (De Bra et al., 2003; Romero et al., 2002), the tool and our methodology can be easily applied to other web-based educational systems, and specifically ASWEs. The only provisions that have to be made are the following ones:

Supplementing usage information. As we explained in more detail in Section 4.1, EPRules needs more usage data than is typically logged by conventional web servers (namely scores obtained in activities and knowledge levels attained). If an ASWE does not generate or log this information, then the teacher can also enter it manually (such as by deriving knowledge levels from the scores of traditional tests or exams).

Combining fine-grained knowledge levels. Some ASWEs use a very fine-grained domain model (Brusilovsky, 2001). In contrast, EPRules only distinguishes concepts (with several scenarios) that are grouped into chapters (Romero et al., 2002). In order to discover rules with EPRules, it will be necessary to combine fine-grained knowledge levels in the given domain model to create the coarse-grained knowledge levels that are used by EPRules.

7. Conclusions

In this article, we 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. We analyzed the different methods (Lavrac et al., 1999; Tan et al., 2002; Yao and Zhong, 1999) that have been proposed to evaluate the quality of the rules obtained by knowledge discovery algorithms, and established the need for multi-objective algorithms instead of classic algorithms that are mono-objective. We 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, the execution times and the percentage of interesting, accurate and comprehensible rules, shows that the algorithms based on Pareto Front (MOGA and especially NSGA) are superior to the other proposed algorithms that only use one evaluation measure or a linear composition of several measures. With regard to the utility of the discovered rules in making decisions about possible modifications in ASWEs, we described the different types of obtained rules and the utility of each type for the improvement of the course. We also presented examples of rules about a Linux course developed with AHA! that our algorithm discovered. Finally, we developed a special tool to help non-experts in data mining carry out the complete rule discovery process. This tool allows the user to carry out the pre-processing of students' usage data, to place restrictions on the types of relationship he wants to discover, to run the data mining algorithms for rule extraction, and to visualize the discovered rules.

We arrived at the following conclusions based on our work:

1. We showed that the use of data mining techniques (in our case prediction rule mining) on usage information generated by ASWEs allows us to discover useful knowledge for improving such systems. These rules can be used by the teacher to make decisions about how to modify the course to improve students' performance.
2. We proposed a methodology to improve ASWEs. However, this methodology can be applied to other types of web-based systems as well since it is domain independent. The only difference is the type of usage information of each particular system.

3. We showed that grammar-based genetic programming algorithms, are very suitable for rule discovery in ASWE. This due to their ability to obtain more interesting and comprehensible rules and to represent the individuals. The teacher can change the individual format by merely modifying the rule codification.
4. We showed that the use of an evolutionary multi-objective approach can improve the obtained results. Concretely, the NSGA approach yields a smaller number of rules with a higher percentage of interest, accuracy and simplicity than the other algorithms.
5. We showed that the discovered rules about 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 the structure and content of the course.

Currently we begin searching for new measures of the subjective interestingness of rules, with the help of education professionals. The teachers will have to decide interactively during the process of rule discovery which are the most interesting rules. Related to this endeavor is the discovery process of hot spot miner (Williams, 1999) in which the individuals are directly evaluated by an expert in each cycle. Perhaps this method may not be applicable to our problem due to the large number of rules that can be obtained in each evolution step. However, a first approach with a small population size may be viable and could show information about measures that model these user preferences for the discovered rules effectively.

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. They would also like to thank the anonymous reviewers and the editor for their professional and careful reading of the paper and for their may valuable detailed comments.

Appendix A. Rule Evaluation Measures

Table A.2 shows the most frequently used rule evaluation measures (Tan et al., 2002; Lavrac et al., 1999) that originated in different areas such as machine learning, data mining, statistics, classification, etc. But almost all of them can be obtained from the contingency table Yao and Zhong (1999), that it is the generalization of the confusion matrix 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 A.3. Table A.1 shows the used symbols and probabilities.

Table A.1. Rule evaluation measures

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

Table A.2. 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

Table A.3. Meaning of the used symbols and probabilities

A:	Instances that match the rule antecedent.
$\neg A$:	Instances that match the negation of the rule antecedent.
C:	Instances that match the rule consequent.
$\neg C$:	Instances that match the negation of the rule consequent.
AC:	Intersection of A and C. Similar definition for $\neg AC, \neg A\neg C$ and $A\neg C$.
n(A):	Cardinality (number of instances) of A. Similar definition for $n(C), n(\neg A)$ and $n(\neg C)$.
N:	The total number of data instances.
p(A):	Relative frequency of A, obtained by $p(A) = \frac{n(A)}{N}$. Similar definition for $p(C), p(\neg A)$ and $p(\neg C)$.
p(AC):	Relative frequency of the intersection of A and C, obtained by $p(AC) = \frac{n(AC)}{N}$. Similar definition for $p(\neg AC), p(\neg A\neg C)$ and $p(A\neg C)$.
p(A/C):	Relative frequency of A conditioned by C, obtained by $p(A/C) = \frac{p(AC)}{p(C)}$. Similar definition for $p(\neg A/C), p(\neg A/\neg C)$ and $p(A/\neg C$.

References

Agrawal, R., Imielinski, T. and Swami, A.: 1993, Mining Association Rules between Sets of Items in Large Databases. In: *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data.* Washington, DC, 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: *Proceedings of Conference on IEEE Systems and Cybernetics.* Tokyo. pp. 940–945.

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

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

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

Bojarczuk, C. C. Lopes, H. S., and Freitas, A. A.: 2001, Constrained-Syntax Genetic Programming for Knowledge Discovery in Medical Databases. In: *10th International Symposium on System Modeling*. Zakopane, Poland.

Brusilovsky, P.: 1998, Adaptive Educational Systems on the World-Wide-Web: A Review of Available Technologies. In: *Proceedings of the Workshop "www-Base Tutoring" at the 4th International Conference on Intelligent Tutoring Systems (ITS 98)*, San Antonio, TX.

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

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

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

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

Delgado, M., Sanchez, D., Martin-Bautista, M.J. and Vila, M.A.: 2001, Mining association Rules with Improved Semantics in Medical Databases. *Artificial Intelligence in Medicine* **21**, 241–245.

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

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

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

Deb, K.: 2001, Multi-Objective Optimization Using Evolutionary Algorithms. *New York, USA: John Wiley & Sons.*

De Bra, P. Wu, H. Aerst, A. and Houben, G.: 2000, Adaptation Control in Adaptive Hypermedia Systems. In: *Proceedings of International 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., and Stash, N.: 2003, AHA! The Adaptive Hypermedia Architecture. In: *Proceedings 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 Conference on E-Learning in Corporate, Government, Healthcare, Higher Education*, Montreal, Canada, pp. 2575–2576

Dubois, P.: 2003, MySQL, Second Edition. Sams, USA.

Fonseca, C.M., and Fleming, P.J.: 1993, Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization. In: *Proceedings 5rd International Conference 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: *Conference on 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: A. Ghosh and S. Tsutsui (eds.): *Advances in Evolutionary Computation*. Springer-Verlag, pp. 819–845

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

Gilbert, R. G., Goodacre, R., Shann, B., Kell, D. B., Taylor, J. and Rowland, J. J.: 1998, Genetic Programming-Based Variable Selection for High-Dimensional Data. In: *Proceedings 3rd Conference Genetic Programming*, San Francisco, CA, USA, pp. 109–115.

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

Han, J., Lakshamanan, L. and Raymond, T. Ng.: 1999, Constraint-based, multidimensional data mining. *IEEE Computer* **32**(8), 46–50.

Heift, T. and 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. and Pompidor, P.: 2002, Evaluating and Revising Courses from Web Resources Educational. In: *International Conference on Intelligent Tutoring Systems*. Biarritz, France, San Sebastian, Spain, pp. 208–218.

Hipp, J., Gntzer, U. and 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.

Jaroszewicz, S. and Simovici, D.: 2002, Pruning Redundant Association Rules Using Maximum Entropy Principle. In: *Proceedings of the 6th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*. Taipei, Taiwan, pp. 135–147.

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

Koza, J. R.: 1992, *Genetic Programming: on the Programming of Computers by Means of Natural Selection.* Cambridge, Massachusetts: MIT Press.

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

Liu, B. and Hsu, W.: 1996, Post-Analysis of Learned Rules. In: *Proceedings of National Conference on Artificial Intelligence*. Portland, Oregon, USA, pp. 828–834.

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

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

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

Minaei-Bidgoli, B. and 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.

Mitra, S., Pal, S. K. and Mitra, P.: 2001, Data mining in soft computing framework: a survey. *IEEE Transaction on Neural Networks* **13**(1), 3–14.

Noda, E., Freitas, A. and Lopes, H. S.: 1999, Discovering Interesting Prediction Rules with a Genetic Algorithm. In: *Proceedings of the Congress on Evolutionary Computation*. Washington DC, USA, pp. 1322–1329.

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

Ortigosa, A. and 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.

Padmanabhan, B. and Tuzhilin, A.: 2000, Small is Beautiful: Discovering the Minimal Set of Unexpected Patterns. In: *Proceedings of the International Conference on Knowledge Discovery and Data Mining.* Boston, MA, USA, pp. 54–64.

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

Piatesky-Shapiro, G. and 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. and 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. *Proceedings of IJCAI*, pp. 304–307.

Ratle, A. and Sebag, M.: 2000, Genetic Programming and Domain Knowledge: Beyond the Limitations of Grammar Guided Machine Discovery. In. *Proceedings of 6th Conference on Parallel problem solving for nature*. Paris, France, pp. 211–220.

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

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

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

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

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

Smythe, P. and 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: *Proceedings of the Knowledge Discovery and Data Mining*. Montreal, Canada, pp. 275–281.

Silverstein, A., Brin, S. and 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. and Deb, K.: 1994, Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation* **2**(3), 217–249.

Srivastava, J., Cooley, R., Deshpnade, M. and 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. and Srivastava, J.: 2002, Selecting the right Interestingness Measures for Association Patterns. In: *Proceedings of the 8th International Conference on Knowledge Discovery and Data Mining*. Edmonton, Canada, pp. 32–41.

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

Ventura, S., Ortiz, D. and 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: *Conference on Educational Multimedia, Hypermedia and Telecommunications*. Denver, Colorado, pp. 2005–2006.

Whigham, P. A.: 1995, Gramatically-Based Genetic Programing. In: *Proceedings 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: *Conference on Knowledge Discovery and Data Mining.* Beijing, China, pp. 184–193.

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

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

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

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

Zaïane, O.R. and Luo, J.: 2001, Towards Evaluating Learners' Behaviour in a Web-Based Distance Learning Environment. In: *Proceedings of the IEEE International Conference on Advanced Learning Technologies*. Madison, Wisconsin, pp. 357–360.

Zaïane, O.R.: 2002, Building a Recommender Agent for e-Learning Systems. In: *Proceedings of the International Conference on Computers in Education,* Auckland, New Zealand, pp. 55–59.

Authors' Vitae

Dr. Cristóbal Romero is an Assistant Professor in the Computer Science Department of the University of Córdoba, Spain. He received his Ph. D. in Computer Science from the University of Granada in 2003. His research interests lie in artificial intelligence in education and data mining.

Dr. Sebastián Ventura is an Associate Professor in the Computer Science Department of the University of Córdoba, Spain. He received his Ph. D. in the Sciences from the University of Córdoba in 1996. His research interests lie in soft-computing and its applications.

Dr. Paul De Bra is a Professor in the Computer Science Department of Eindhoven University of Technology in the Netherlands. He received his Ph. D. in Computer Science from the University of Antwerp in 1987. His research interests lie in adaptive hypermedia systems and web-based information systems.