Using a hybrid A.I. approach for exercise difficulty level adaptation

Constantinos Koutsojannis, Grigorios Beligiannis, Ioannis Hatzilygeroudis and Constantinos Papavlasopoulos
Department of Computer Engineering & Informatics, School of Engineering
University of Patras, Hellas (Greece)
E-mails:ckoutsog@ceid.upatras.gr, beligian@ceid.upatras.gr, ihatz@ceid.upatras.gr, papavlas@ceid.upatras.gr

Jim Prentzas
Department of Informatics and Computer Technology
Technological Educational Institute of Lamia, Hellas (Greece)
E-mail:dprentzas@teilam.gr

Abstract: In this paper, we present an intelligent and adaptive web-based education system that uses a hybrid AI approach for determination of the difficulty levels of the provided exercises. More specifically, a combination of the expert systems approach and a genetic algorithm approach is used. A genetic algorithm is used to extract some kind of rules from the data acquired from the interactions of the students with the system when answering to questions/exercises. Those rules are used to modify expert rules provided by the Tutor. In this way, feedback from the students is taken into account for determination of the difficulty levels of the questions/exercises. This is important because the difficulty levels of the exercises are taken into account for the evaluation of the knowledge levels of the students with regards to various concepts. Experimental results show that a significant part of questions/exercises may need to change their level of difficulty. Furthermore, the validity of the method is experimentally showed.

Keywords: Intelligent Web-Based Education, Intelligent E-Learning, Exercise adaptation, Expert systems, Genetic Algorithms, Hybrid Intelligent Systems

Biographical notes: Dr Constantinos Koutsojannis holds a PhD in Medical Physics and studies for a PhD in Artificial Intelligence at the Department of Computer Engineering and Informatics, University of Patras, Greece. Special areas of interest are intelligent web based educational systems and medical expert systems. Dr Koutsojannis is currently member of the teaching staff at the Health Science Department of Technological Educational Institute of Patras, Greece.

Dr Grigorios Beligiannis received his PhD from the Department of Computer Engineering and Informatics, University of Patras, Greece, in 2002. Since then he is working as a post-graduate researcher at the Pattern Recognition Laboratory at the above Department. He has published several papers concerning the application of evolutionary algorithms in many real-world problems (system identification and parameter estimation of linear and nonlinear systems, text classification, user/student profile optimization, biomedical data processing and classification). His current research interests are in evolutionary and genetic programming, hybrid intelligent systems, data mining, system analysis and detection, and bioinformatics.

Author

Dr Ioannis Hatzilygeroudis is currently an Assistant Professor at the Department of Computer Engineering & Informatics, University of Patras. He is the Associate Editor-in-Chief of the Inetrnational Journal of Artificial Intelligence Tools (IJAIT) and member of the Editorial Boards of the International Journal of Hybrid Intelligent Systems (IJHIS), the International Journal of Web-Based Communities (IJWBC) and the International Journal of Computational Intelligence (IJCI). He has also been member of the PCs of several AI related conferences (e.g. IEEE ICTAI, FLAIRS) for a number of years. He has organized a number of special/invited sessions in those and other conferences. His main research interests are: artificial intelligence, hybrid knowledge representation, expert systems, knowledge engineering and intelligent educational systems. He has published over 50 papers. He is member of IEEE, ACM, AAAI and currently the Vice President of EETN (the Hellenic AI Society).

Mr Constantinos Papavlasopoulos has graduated from the Department of Computer Engineering & Informatics, University of Patras, Greece, in 2005. He is currently an MSc student at the same Department. His areas of interest are web based educational systems and artificial intelligence.

Dr Jim Prentzas received his PhD from the Department of Computer Engineering & Informatics, University of Patras, Greece, in 2002. He is currently member of the teaching staff at the Department of Informatics & Computer Technology, Technological Educational Institute of Lamia, Greece. He has been member of the PC of the FLAIRS conference for some years. He has participated in a number of National and European research projects. His main research interests are: artificial intelligence, intelligent tutoring systems, knowledge representation, web applications and geographical information systems. He has published over 30 papers.

1 Introduction

There have been two popular categories of educational systems: a) Intelligent Tutoring Systems (ITSs) and b) Adaptive Educational Hypermedia Systems (AEHSs).

ITSs take into account the user's preferences and knowledge level and adapt presentation of the teaching material to his/her needs. This is mainly achieved by using AI techniques to represent pedagogical decisions as well as domain knowledge and information regarding each student (Polson and Richardson, 1988). ITSs were usually developed as stand-alone systems. However, the emergence of the WWW gave rise to a number of Web-based ITSs, a type of Web-Based Intelligent Educational Systems (WBIESs) (Brusilovski and Paylo, 2003).

AEHSs are specifically developed for hypertext environments such as the WWW. The main services offered to their users are adaptive presentation of the teaching content and adaptive navigation by adapting the page hyperlinks (Brusilovski, 1998). Compared to 'classical' ITSs, they offer a greater sense of freedom to the user, since they allow a guided navigation to the user-adapted educational pages. Furthermore, they dynamically construct or adapt the educational pages in contrast to 'classical' ITSs in which the contents of the educational pages are typically static. Enhancing AEHSs with aspects and techniques from ITSs creates a type of what are called Adaptive and Intelligent Educational Systems (AIESs) (Brusilovski and Paylo, 2003).

On the other hand, e-learning environments provide facilities mainly for helping course generation and management and refer to both the tutors and the students. Adding facilities (intelligent or not) for tutors in WBIESs make them a kind of intelligent e-

learning systems (IELSs) (e.g. Christea et al, 2004). An important function of such systems is management of exercises, which provides facilities for adapting exercises to the students and/or generating exercises, thus helping the tutor. Existing systems dealing with exercises usually concern automatic generation of exercises adapted to the students or adaptation of the content of the exercises. For example, in Multibook, exercises can be automatically created based on an ontology representing the curriculum of a course (Fischer and Steinmetz, 2000). In (Rioja et al, 2003) parametric type exercises are generated and sequenced according to the students needs. A parametric exercise is an exercise that can be replicated as many times as necessary with different data. Those replications can be of variable complexity, therefore exercises sequencing is of importance. Finally, in (Lahtinen and Sutinen, 2004) exercises are constructed according to the student type based on 'exerciselets', which are modular components of complete exercises. An exercise is constructed based on vector space distance between the student and corresponding exerciselets.

However, an aspect that has not been paid attention to is the determination of the difficulty level of testing questions or exercises. Given that there are systems that use the difficulty level of questions or exercises for student evaluation (Hatzilygeroudis eta al, 2005a; Hatzilygeroudis eta al, 2005b) determination of the right difficulty level can be important. Also, adaptation of the difficulty levels to specific students may be useful. We are not aware of any attempt that deals with that aspect. Also, a few of the existing systems integrate more than one AI technique to implement exercises management or adaptation processes.

To help the students and the tutors in our Department, we constructed an Artificial Intelligence Teaching System (AITS) to assist learning and teaching in the context of the course of "Artificial Intelligence". AITS is an adaptive and intelligent system. It adapts the course material to the student's needs as much as possible, based on his/her profile and knowledge level [8, 9]. Knowledge level is evaluated via an expert system, which takes into account the difficulty level of questions/exercises. Additionally the system provides means to the tutor for constructing questions and tests in a structured way (Hatzilygeroudis eta al, 2005a).

In this paper, we present a new capability of the system. With the help of a hybrid intelligent system, the system adapts the difficulty level of exercises, taking into account the number of tries, the number of hints and the time spent on exercises by the students. Thus, by incorporating AI techniques it improves tutors' adaptability in a dynamic and quantifiable way enriching its importance for contemporary education in our Department.

The structure of the paper is as follows. Section 2 presents the system architecture. In Section 3, the domain knowledge structure is described. Section 4 deals with the learning process, whereas Section 5 with the exercises management tool of the system. Section 6 briefly refers to student evaluation. Section 7 presents exercises adaptation and how the hybrid intelligent system is involved in that. Section 8 deals with implementation issues and finally Section 9 concludes the paper.

2 System Architecture

The architecture of the system is depicted in Figure 1. The system consists of seven units: the User Interface (UI), the Student Modeling Unit (SM), the Tutoring Unit (TU), the

Evaluation Unit (EU), the Exercise-Adaptation unit (EA), the Intelligent Unit (IU) and the Exercises database (EDB).

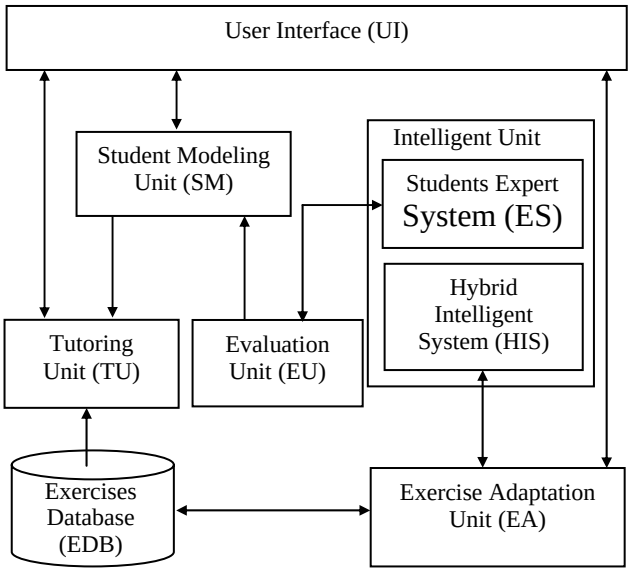
Through UI the student initially subscribes to the system. During subscription some personal settings are saved. After subscription the student can, at any time, enter the system through the UI.

SM contains all the information about students, like their preferences, interests, knowledge level, etc. We use stereotypes to acquire an initial profile for each student, so the system can use that profile for adaptation. This is achieved by using a questionnaire that gets information about student's knowledge, preferences and goals. This information is saved in the SM, which can later be updated by acquiring new information.

TU is responsible for the teaching process. A student can select a learning goal from the learning goals tree. The corresponding material is then presented to the student. The material consists of theory and examples. The student is also able to select a test to evaluate him/herself about how well he/she has learned a concept so far. After the student has taken a test, the system stores the results concerning each concept, which the student has been examined at, and, if the results are not satisfactory, it advises him to study again the corresponding concepts and suggests some new tests that contain questions only about the problematic concepts.

The main goal of EU is to evaluate student's progress due to his/her interaction with the system. This evaluation is achieved through testing. From testing results the tutor is able to watch each student's progress. He/she is also able to see some statistical results over the concepts and sections the student has been examined at. The system can also provide the overall results to the tutor, about all students that were examined at a specific test. The Students Expert System (SES) decides upon the knowledge level of a student.

Figure 1 System Architecture



EA has as main goal to (re)evaluate the exercises' levels of difficulty and thus improve system's adaptivity. If the initial difficulty level of an exercise has been proved not to be the appropriate, after a number of users' entries, the unit changes the corresponding level. This is achieved by a hybrid intelligent system that uses the combination of an expert system and a genetic algorithm, which takes into account the interactions of the students with the system.

The IU consists of two intelligent systems. The one is an expert systems (SES) and deals with student evaluation. The other, the Hybrid Intelligent System (HIS), deals with adaptation of questions/exercises, which are stored in EDB.

3 Domain knowledge

The domain knowledge of the system, at the moment, concerns fundamental aspects of knowledge representation. Domain knowledge is structured in a tree-like way. The root of the tree is the above subject. The subject is divided in sections and the sections into sub-sections. Each sub-section deals with a number of concepts, which are the leaves of the tree. Subsections may have common concepts.

The domain knowledge tree, described above, is displayed as far as the subsections level at the navigation area, at the left side of the user interface. From that tree the student can choose a learning goal (subsection). Each subsection corresponds to a *learning page*, which is an asp page. That is, only subsections correspond to displayable material. The learning page of the selected subsection is currently presented in the content area. Each learning page deals with a number of *concepts*. More specifically, it contains an ordered list of concepts. Each concept is linked to the corresponding concept page. Concept pages constitute the real teaching material.

The teaching material, apart from concept pages, however, includes all the available questions/exercises, which are stored in the exercises database (EDB) (see Fig. 1) and are used for the creation of the tests. EDB also contains the user interaction data for each learning page, used for exercises difficulty level adaptation (see Section 7). Each learning page is associated with a *learning test*. Each test consists of a number of exercises that refer to the concepts of the associated learning page. An exercise may be a multiple choice question or something more complex (e.g. to apply a process). Also, each concept page offers a test concerning the concept.

4 Learning process

The learning method (implicitly followed) is based on the traditional theory-examples-exercises paradigm (although the user can follow his own method). That is, for each topic, the theory is first presented. Then, some examples are given. Finally, the student is called to do some exercises. Theory consists in presenting a number of concepts. Those concepts are presented in a simple-to-complex way. That is, the simple concepts are presented first and the complex concepts (that require the knowledge of one or more simpler concepts) are presented afterwards. This is depicted in the ordered list.

Furthermore, the student can review a previous concept at any time. He/She is also not forced to follow the system's way of teaching, but can make his/her own choices for

studying a learning page. For example he can jump to complex concepts without taking a look at simpler ones. In many concept pages there are links to other concepts that are prerequisite to the concept of the page. So, the student, if needed, can recall the theory about the prerequisite concepts. After having looked at the recalled theory, he/she can return back to where he/she was before and go on with his/her studying.

The student can check his/her knowledge level after having studied a learning page, by doing the test specified at the end that page. Any time a student has finished the study of a particular learning page, he/she can take an appropriate test. The questions of the test are presented one at a time. The student can answer each question independently and in his own sequence. In case the student gives a wrong answer to a question, he/she can try again. Before that, he/she may either see a hint or the correct answer. Doing any of those, the mark that will be assigned to the student for that question is affected. The systems collects from each user the number of hints for each exercise, the total time that he/she spent to answer it correctly and the number of tries. Each exercise's difficulty level is determined by all the above information and the nature of corresponding concept according to tutors' experience.

5 Exercises management

The system offers to the tutor the capability of domain knowledge and exercise management. The tutor can deal with management of the domain tree and the teaching material (learning pages, concept pages, questions/exercises and tests). Management of the domain tree concerns the insertion, deletion or change of a sub-tree of the domain tree. The tutor can insert a new sub-tree at any level of the tree (chapter-sections-subsections-concepts, section-subsections-concepts or subsection-concepts) or delete an existing sub-tree at any level or modify one. In other words, the tutor can change the domain knowledge tree at any time according to his/her teaching needs.

The most important part of teaching material management is the management of the questions/exercises. An exercise is considered as a structured object that consists of the following parts:

- its name
- its body
- the concept it concerns
- its difficulty level (1-5),
- its possible answers (1 to 4)
- one or more hints

Every possible answer is assigned a mark and an explanation. The mark represents the student's expected mastery degree upon the concept, if the corresponding answer is selected. An explanation provides information about why the corresponding answer is wrong or right. The hint is a kind of clue to help a student to answer the question correctly, or to find the appropriate concept to study again.

The tutor is able to insert, delete or modify all the parts contained in a question. To insert a new question the tutor has to select the corresponding form (tutors' area). In that form the tutor has to fill in all the above mentioned parts of the question. Every question inserted is referred to one of the existing concepts. The tutor has to know the chapter, the section and the subsection that concept belongs to. Questions that refer to the same concept may have different difficulty levels.

To delete a question, the tutor has to select the corresponding question and then click a delete button. Modification of a question requires first the selection of the question. Then the tutor can update any part of the question. For example, the tutor can add a new possible answer or remove one or modify existing ones. Also, the tutor can see all the questions and all information concerning them.

Another facility that is available to the tutor is the management of tests. All available tests can be shown in a list. The content of each test can be displayed by selecting the test's name. In case that the tutor wants to delete one or more tests, he/she can check them in the list and then click the delete button. To create a new test, the tutor has first to give a name for this test. Then, through an appropriate form, he/she selects the chapter, the section, the subsection and the concept as well as the difficulty level and number of the questions that are going to be inserted in the test. Questions that satisfy the above criteria are randomly selected and then inserted in the test. The tutor can add as many questions as he/she wishes, following the same procedure.

In the corresponding form some boxes are filled in an automated way. Every time the tutor selects a chapter, the sections contained in that chapter are shown. Also, for each selected section, its subsections are automatically recalled and shown.

Finally, new concept and learning pages can be added to or deleted form the system. Also, the tutor can modify existing pages.

6 Student evaluation

In web-based educational systems, one of the most important functions is adaptation. Adaptation has mainly to do with the learning content presented to the user. One of the most important functions of an adaptive system, which is crucial for the specification of the appropriate learning content, is student evaluation. Student evaluation refers to the evaluation of the knowledge level of a student after having dealt with a learning page. In other words, how well a student has learnt the concepts of a learning page.

This is achieved by processing the results of the exercises offered at the end of a learning page. Student evaluation is important for both the student and the tutor. A student can be evaluated at two levels: (a) the concept-level and (b) the topic-level. The concept-level evaluation deals with the level of understanding of the concepts of a learning page test, whereas the topic-level evaluation deals with the level of understanding of the topic of a learning page, i.e. the test as a whole. The knowledge level of a student, as far as both a concept and a topic are concerned, is classified in one of the following five categories: (a) excellent, (b) very good, (c) good, (d) average and (e) low.

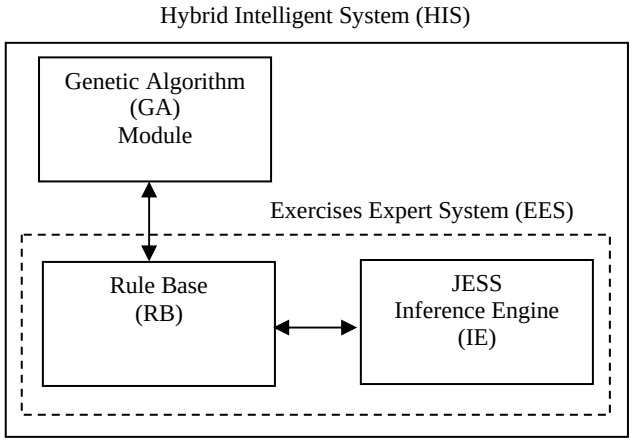
In AITS, student evaluation is achieved with the help of a rule-based expert system (SES), which is part of IU (see Fig. 1). The test results are passed to the SES, which decides about the knowledge levels of the students for the associated concepts. Details of this process are presented in (Hatzilygeroudis et al, 2005a) (an alternative process is presented in (Hatzilygeroudis et al, 2005b)).

7 Exercises adaptation

Author

Estimation of the knowledge level of a concept is based, among others, on the difficulty level of the correctly answered exercises included in the test. So, the right determination of the difficulty level of an exercise is important. This is mainly done by the tutor. The tutor specifies the difficulty level of each exercise, based on the complexity of the exercise and his/her experience. Exercises are classified in five levels of difficulty (1 to 5). However, classification of exercises made by the tutor is not always correct. This may lead to incorrect knowledge level estimations. Also, he/she doesn't (or it is difficult to) take into account any feedback from the students.

Figure 2 The Hybrid Intelligent System Structure



In AITS, the difficulty levels of the exercises can be (re)evaluated based on the performance of the students. The tutor can use an optional tool to (re)evaluate the difficulty levels of all the exercises in the database using a single button in tutors' interface. This is achieved by EA unit and HIS. The EA unit calls HIS to do the main work.

7.1 The hybrid intelligent system

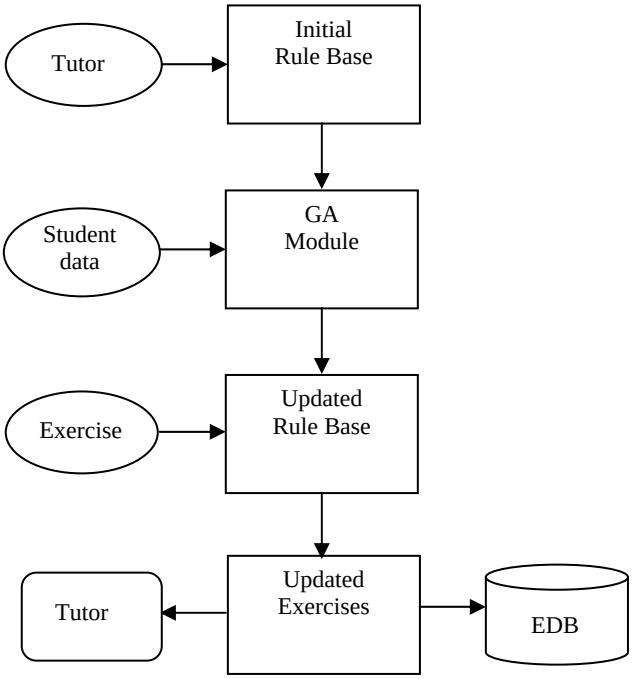
HIS consist of three components: the Rule Base (RB), the JESS Inference Engine (IE) and the Genetic Algorithm (GA) Module (see Figure 2). The RB contains a number of rules that are used for the evaluation of the difficulty levels of the exercises. The JESS IE is the inference mechanism that applies the rules and produces the new difficulty levels. Finally, the GA Module update the rules in the RB based on a genetic algorithm and the data collected from the interactions of the students with the system as far as exercises are concerned. This process is depicted in Figure 3 and explained in the next subsection in some detail.

The RB and JESS IE constitute the Exercises Expert System (EES). EES is implemented in the Jess, a Java based expert systems shell (Friedman-Hill, 2003).

7.2 The adaptation process

Initially, the tutor sets the difficulty levels of the exercises (integers between 1 and 5), based on the nature of the exercises and his/her experience. Alongside the difficulty level, the tutor also gives a range of difficulty for each exercise, i.e. the minimum and maximum possible difficulty that can be assigned to the exercise.

Figure 3 The process of exercises adaptation



At the same time, the tutor is called to give a number of initial rules. Those rules represent the way the tutor would estimate the difficulty levels of the exercises based on available data from students' interaction with the system. In other words, those rules are an explicit representation of what the tutor implicitly (or intuitively) does based on his/her experience. There are two reasons for doing that, a technical and an educational. On the one hand, the GA Module needs an initial rule base to produce an updated one. On the other hand, the initial rules can be used to specify the difficulty levels of the exercises after having a number of student data during a testing phase.

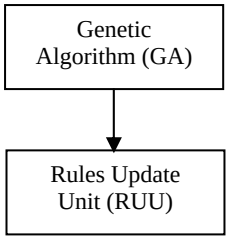
That data includes information about the number of tries the students did, the number of hints they used, the time they spent and whether their answer was correct or not for each exercise. In the above, by "number" we mean the average value of the corresponding parameter.

All the above are done in an off-line mode. During the use of the system, when a sufficient amount of data has been gathered, the system proposes to the tutor to trigger the process for the exercises (re-)adaptation. If accepted by the Tutor, the GA Module takes over. The GA module uses a GA approach to evolve an updated rule base from the old one, based on the students' interaction data. How this is done is described in the next subsection.

After that, the updated rule base is used via the JESS IE to specify the new difficulty levels, if any, and produce the updated exercises. The process can be repeated every time an adequate number of students have used the exercises under consideration.

Let's look at the following example. The tutor inserts some new exercises with difficulty level 2. According to his/her experience he/she initially creates a rule (in the initial rule base) that gives this difficulty level to an exercise when a student needs between 2 and 5 minutes to answer it, having used one or no hints and having made 2 or less tries. Additionally, the tutor gives the minimum and the maximum difficulty level values for each exercise. After the use of the system for a sufficient period the system uses the data from the interactions of all the students that have tried to answer those exercises and evolves a new rule (with the use of GA module). The system compares the initial with the evolved rule and if the new rule is different from the initial one, because, for example, the mean time that the students needed to answer the exercise was significantly over 5 minutes, the new rule is inserted in the updated rule base. Then the difficulty levels of the exercises are re-evaluated and may change. If a proposed new level is greater than the specified maximum value, the adapted value will become equal to the maximum one. Similarly, if a proposed new level is less than the specified minimum value, the adapted value will become equal to the minimum one. Those limits assure that difficulty levels do not solely depend on the background and skills of the students at some period. So, a group with lower/higher background and skills would not affect in a free way, but in a controlled one.

Figure 4 Structure of the GA Module



This kind of adaptation takes into account the tutors' expertise as well as the students' feedback. Thus, difficulty levels are adapted to both factors, the tutor and the students. The tutor is able to see the proposed changes.

7.3 The GA module

The basic structure of the GA module is depicted in Figure 4. The GA Module consists of the GA and the rules update unit (RUU).

GAs, in general, operate on binary string structures, analogous to biological creatures (genomes). These structures are evolving in time according to the rule of survival of the fittest, by using a randomized, yet structured, information exchange scheme. Thus, in every generation, a new set of binary strings is created, using parts of the fittest members of the old set (Mitchel, 1996; Michalewitcz, 1999). GAs process a binary coding of the parameter space and work on it. This coding (which is an essential part of the GA design procedure) results in formation of binary strings.

Figure 5 The structure of the genome

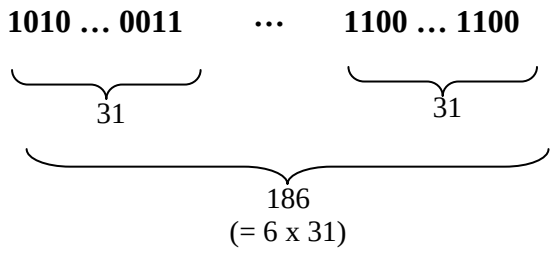
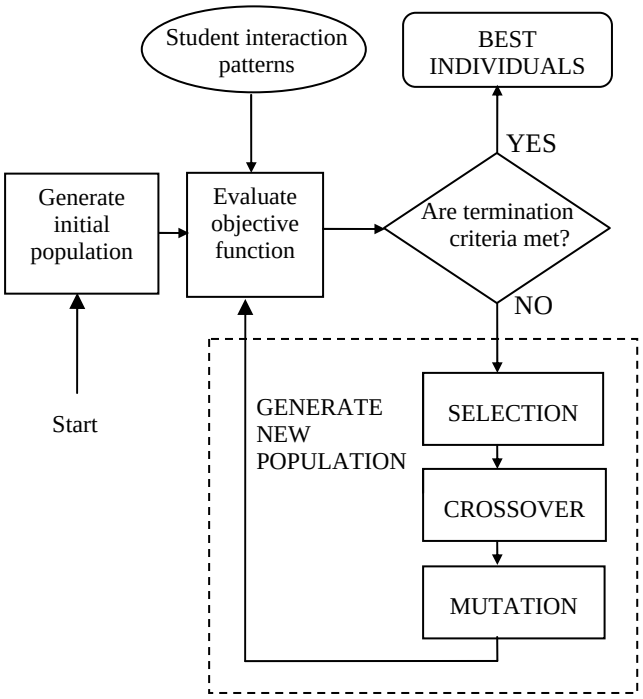


Figure 6 The Genetic Algorithm



In our case, the coding scheme comprises 6 input parameters: T1, T2, E1, E2, H1, H2. T1 and T2 represent the lower and upper limits of a time interval, E1 and E2 the lower and upper limits of a tries interval and H1 and H2 the lower and upper limits of a hints interval. Each parameter is coded into a binary string using 31 bits (see Figure 5). Each such binary string constitutes a gene of the genome of the GA. Thus, the length of the genome is 186 (= 6 x 31) bits. This binary string (the genome of the GA) will be evolved.

The steps of the GA are presented in Figure 6. The genetic algorithm used is a classical one. The initial population is generated using a set of random binary strings (genomes).

The objective function estimates the goodness (fitness) of each individual (genome) in the population. The fitness of each individual is based on how many student interaction patterns it can satisfy. A student interaction pattern has the following structure,

<time tries hints level>

that represents how much time the student spent on an exercise, how many tries he did, how many hints he asked for and what was the difficulty level of the exercise. A pattern is satisfied if 'time' belongs to [T1, T2], 'tries' belongs to [E1, E2] and 'hints' belongs to [H1, H2]. More specifically, the fitness of a genome is specified by the algorithm in Fig. 7. GA applies successively to student patterns referring to exercises with difficulty level 1, 2, …, 5.

Figure 7 Algorithm for fitness computation

```
for each student interaction pattern do
    fitness = 0;
    for each genome do
        if T1<time
          then fitness = fitness + 1;
        if T2>time
          then fitness =fitness + 1;
        if (T1<time) and (T2>time)
          then fitness = fitness + 1/(T2-T1);
        if (E1<tries) and (E2>tries)
          then fitness = fitness + 1/(E2-E1);
        if (H1<hints) and (H2>hints)
          then fitness = fitness + 1/(H2-H1);
```

The termination criterion is the number of generations, that is, the evolution procedure is terminated when a specific number of generations (in our case 10) are completed. While the termination criterion is not met, the whole evolution procedure is taking place for the current population, that is, selection, crossover and mutation are performed (Mitchel, 1996). When the termination criterion is met, GA evolution procedure is terminated. Then, the best individual of the resulted population for the corresponding difficulty level is selected, let it be <T1*, T2*, E1*, E2*, H1*, H2*>. This represents the shortest possible intervals for time, tries and hints for the specific difficulty level that satisfy most of the student interaction patterns. This can be translated into a rule of the form

```
if (T1*<= time <= T2*) and
   (E1*<= tries <=E2*) and
   (H1*<= hints <=H2*)
then difficulty_level = levelx
```

where 'levelx' is the difficulty level the student patterns refer to.

The RUU compares each such produced rule with the corresponding existing rule (provided by the Tutor) and decides on whether and how the existing rule will be modified. After that it updates the rule base.

Finally, the updated rule base is used to calculate the new difficulty levels of each exercise that will be further available in the updated exercise database. In doing that, the average values of the parameters of the student patterns related to a difficulty level are used.

7.4 Exercises statistics

The system also provides a number of statistics related to the exercises. Those statistical results are very helpful for new exercise entries and after the end of each semester for curriculum adaptation according to the total students performance.

The statistics provided to the tutor are the following:

- the total number of the students for each exercise that answered it correctly
- the percentage of the correct initial exercise classifications
- the percentage of the modified exercises classifications
- the final total number of exercises per difficulty level per concept
- the students' performance according to the exercise difficulty levels

Thus, the system provides to the tutor a useful tool to check the quality of each exercise that he/she has inserted into the system and each test that he/she has designed for student evaluation. Using that information the new exercises will be better classified by the tutor and thus better helping students at every part of the educational context, because the exercise material has been adapted to their needs.

7.5 Experimentation

We applied the above process to students' data from the current academic year. We performed a two-phase experiment to evaluate our proposal. Initially, 40 students used the system and dealt with the questions/exercises. We used a set of 36 multiple-choice questions of all levels of difficulty. The students were instructed to answer the questions following some guidelines, like: be concentrated, don't waste time for resting, don't use the hints when not necessary, don't answer questions simply by chance.

The initial rules set by the tutor are presented in Table 1. They are relatively simple rules used as a basis for adaptation of difficulty levels.

The rules resulted from the GA are presented in Table 2. Notice that no rules are the same. The differences concern all three parameters, but mainly the mean time spent at each category of exercises.

In the second phase, the same set of exercises was used under the same conditions by a second group of 20 students, after adaptation of their difficulty levels based on the rules of Table 2. The rules resulted from the GA are presented in Table 3. Notice that only the rules concerning categories (levels) 4 and 5 have been differentiated as far as the number of hints (level 4) and the time spent (both levels) are concerned.

Given the above results, 27 out of 36 questions/exercises had to change their difficulty levels values, after the first phase. After the second phase, only 6 out of 36 questions had to change level values.

Author

This two-phase experiment constitutes a kind of a proof for our method validity. Indeed, given the changes made to the difficulty levels after the first phase (Table 2), the changes resulted after the second phase (Table 3) were very limited. This means that the difficulty levels were realistically specified after the first phase.

Table 1 Tutor's rules for difficulty level determination

A/A	Tries	Hints	Time (min)	Difficulty Level
1	<= 2	1	< 2	1
2	<= 2	1	< 3	2
3	<= 2	<=2	< 4	3
4	<= 2	<=2	< 5	4
5	<= 3	1	< 5	4
6	<= 2	<=2	> 5	5
7	<= 3	1	> 5	5

Table 2 Rules resulted from the 1st application of GA.

A/A	Tries	Hints	Time (min)	Difficulty Level
1	<= 1	0	<= 0,5	1
2	<= 1	0	< 1	2
3	<= 1	0	< 2	3
4	<= 2	<=2	2 - 4	4
5	<= 2	<=2	> 4	5

Table 3 Rules resulted from the 2nd application of GA

A/A	Tries	Hints	Time (min)	Difficulty Level
1	<= 1	0	<= 0,5	1
2	<= 1	0	< 1	2
3	<= 1	0	< 2	3
4	<= 2	<=1	2 - 3	4
5	<= 2	<=2	> 3	5

A problem here is the following. Given that we choose the best individual each time, we can have only one rule for each difficulty level. This is not realistic, since it cannot cover all cases. Another problem following the previous one is, given that the new rules should replace some of the old ones, which rules of Table 1 will be replaced. This is easy in cases where one rule exists for a difficulty level in Table 1 (e.g. rule 1). Also, it is easy in the cases like that of level 5. Rule 5 in Table 2 will replace rule 6 in Table 1 (because it is clear that they are closer). However, in the case of level 4 it is difficult to choose. At the moment we choose one of them in random. However, a more sophisticated solution is needed at this point, which is a function of the RUU.

Another point, which is worth of mentioning, is the fact that the rules may not be te same for all same level questions. For example, multiple-choice questions usually take less time than questions that refer to application of a process, although they can be of the same difficulty. So, different rules are needed for different types of questions for the same difficulty level.

8 Implementation issues

The developed system is web-based, so it works only through the WWW and cannot be downloaded and function at the student's computer. It is implemented in ASP. By using ASP, we've added a dynamic functionality to the system, which gives it the ability to interact with the student and the tutor. Usage of ASP can cause no problems in the communication with the database, which was built with MYSQL.

Another important implementation issue was that of the EES, used for exercises adaptation. EES is implemented in Jess, an expert system cell implemented in Java (Friedman-Hill, 2003). After a student has finished a test, a fact for each individual question of the test is saved in a file. Then SES, which is also implemented in Jess, is called to evaluate the student's knowledge levels. Additionally, each time that the tutor triggers the exercise adaptation procedure, HIS is called to re-evaluate the difficulty levels of the exercises. This is achieved with the help of the appropriate ASP commands that give the opportunity of executing a command line program. The expert systems, after taking the facts from a text file, return the results in another text file.

The type of GA used in the EC module of HES is the classic simple GA (Vose, 1998). The representation used for the genomes of the genetic population is the classic binary string. As far as the reproduction operator is concerned, the classic biased roulette wheel selection is used. The crossover operator used is uniform crossover (with crossover probability equal to 0.9), while the mutation operator is the flip mutator (with mutation probability equal to 0.001). Except of that, the size of the population is set to 50 while the GA uses linear scaling and elitism (Gen and Cheng, 1997).

The GA is implemented using the C++ Library of Genetic Algorithms GAlib (http://lancet.mit.edu/ga/) and especially the GASimpleGA class for the implementation of the GA (non-overlapping populations) and the GABin2DecGenome class for the binary string genomes (an implementation of the traditional method for converting binary strings to decimal values). All the experiments were carried out on an Intel Pentium IV 2.7GHz PC with 256 MB RAM.

9 Conclusion and discussion

In this paper, we present an intelligent and adaptive web-based education system enhanced with e-learning system facilities. We concentrate on the determination of the difficulty levels of the questions/exercises provided to the students for testing purposes. A hybrid AI approach is used to achieve that, by taking into account the feedback from the interactions of the students with the system. Two intelligent components work on- and off-line to improve the exercises, which have been inserted in the system by the tutors, aiming at the best possible educational result in the wide area of AI education in our Department.

More specifically, a combination of the expert systems approach and a genetic algorithm approach is used. A genetic algorithm is used to extract some kind of rules from the data acquired from the interaction of the students with the system when answering to questions/exercises. These rules are used to modify expert rules provided by the Tutor. In this way, feedback from the students is taken into account for determination of the difficulty levels of the questions/exercises. This is important because the difficulty levels of the exercises are taken into account for the evaluation of the knowledge levels of the students with regards to various concepts. Experimental results show that a significant part of questions/exercises may need to change their level of difficulty.

This facility can be exploited in various ways by the tutor:

- A more realistic classification of the questions/exercises can be achieved. This may have a positive impact on students' learning.
- It could be easily extended to be used for a personalized adaptation of the difficulty levels (i.e. adaptation to each student individually).
- The tutor can use it to specify the difficulty levels of various types of quite similar questions/exercises during a training period and use the resulted levels for questions of the same type.
- The tutor can re-estimate those levels from time to time.
- A careful study of the proposed changes by the system can result in useful ideas for improving the questions/exercises and/or the teaching material.

Apart from the above, the system provides useful statistics related to the exercises. Thus, it integrates facilities met in e-learning systems. This can also provide useful information for curriculum improvement.

However, there are some points that the system can be improved at. First, the extraction of rules via the GA is fairly simple at the moment. For example, we always choose the best (fittest) individual at the end of the process. Thus, we produce only one rule for classifying each difficulty level. This may be adequate, if it satisfies a large percentage (how much?) of the student interaction patterns, otherwise may not and needs improvement. One solution would be then not to rely only on the best individual, but to the second and possibly the third better individual, depending on the pattern satisfaction percentages or more sophisticated metrics. Alternatively, we could use more than one best individual taken from different runs of the algorithm, if there are different such individuals. This is a direction for further research.

Given that, for example, time limits are not really clear-cuts, another direction for further work could be the use of Fuzzy Logic in our expert system component to produce more realistic adaptation as well as to provide personalized adaptivity facilities.

Acknowledgements

We thank the European Social Fund (ESF), Operational Program for Educational and Vocational Training II (EPEAEK II), and particularly Program PYTHAGORAS for funding this work.

References

Brusilovsky, P. and Paylo, C. (2003) Adaptive and Intelligent Web-Based Educational Systems, *International Journal of AI in Education 13*, pp. 156-169.

Brusilovsky, P. (1998) Methods and Techniques of Adaptive Hypermedia. In: Brusilovsky, P., Kobsa, A. & Vassileva, J. (Eds.): *Adaptive Hypertext and Hypermedia*. Kluwer Academic Publishers.

Christea, P. D., Tuduce, R., Savescu, I. A., Grogorin, C. A., Tomozei, D.-C., Gradinescu, V. R. and Rangu, C. M. (2004) Prototype Implementation of an Intelligent E-Learning System, *Proceedings of the IASTED International Conference on Web-Based Education (WBE-04)*, Febr. 16-18, Innsbruck, Austria, Acta Press, pp 441-446.

Fischer, S and Steinmetz, R. (2000) Automatic Creation of Exercises in Adaptive Hypermedia Learning Systems, *Proceedings of the 11th ACM Conference on Hypertext and Hypemedia (HT-2000)*, San Antonio, TX, USA, pp. 49-55.

Gen, M. and Cheng, R. (1997) *Genetic Algorithms and Engineering Design*, John Wiley & Sons.

Friedman-Hill, E. (2003) J*ess in Action: Rule-Based Systems in Java*, Manning Publishing.

Hatzilygeroudis, I., Chountis, P., Giannoulis, Ch. and Koutsojannis, C. (2005a) Using Expert Systems Technology for Student Evaluation in a Web Based Educational System, *Proceedings of the IASTED International Conference on Web-Based Education (WBE-2005)*, Feb. 21-23, Grindelwald, Switzerland, pp. 534-539.

Hatzilygeroudis, I., Giannoulis, C. and Koutsojannis, C. (2005b) Combining Expert Systems and Adaptive Hypermedia Technologies in a Web Based Educational System, *Proceedings of the IEEE ICALT-2005*, Kaohsiung,Taiwan, July 5-8 pp. 249-253.

Lahtinen, S.-P. and Sutinen, E. (2004) The Exerciselet Framework in Exercise Adaptation in Web-Based Training, *Proceedings of the IASTED Inetrantional Conference on Web-Based Education (WBE-04)*, Innsbruck, Austria, Feb. 16-18, pp. 192-199.

Mitchel, M. (1996) *An Introduction to Genetic Algorithms*, MIT Press.

Michalewicz, Z. (1999) *Genetic Algorithms + Data Structures = Evolution Programs*, Springer-Verlag, Berlin.

Polson, M.C. and Richardson, J. J. (1988) *Foundations of Intelligent Tutoring Systems*, Lawrence Erlbaum Associates.

Rioja, G.R., Santos, S.G., Pardo A. and Kloos C.D. (2003) A Parametric Exercise Based Tutoring System, *Proceedings of the 33rd ASEE/IEEE Frontiers in Education Conference*, November 5-8, Boulder, CO, S1B 20-26.

Vose, M. D. (1998) The Simple Genetic Algorithm: Foundations and Theory, *MIT Press*.