A Self-Motivating Adaptive Learning System

W. Hämäläinen and N. Myller
Department of Computer Science, University of Joensuu, P.O. Box 111, FIN-80101 Joensuu, Finland

J. Lopez-Cuadrado
Department of Computer Languages and Systems, University of the Basque Country, Aptdo. 649, San Sebastian 20080 Guipuzcoa, Spain

S. Pitkänen
Educational Technology Centre, University of Joensuu, P.O. Box 111, FIN-80101 Joensuu, Finland

whamalai@cs.joensuu.fi

Abstract

We present a new idea how to develop a learning environment that could motivate students to keep on learning. We introduce the idea of learning environment that supports learning by gaming but still gives students the possibility of collaboration. We explain the generation of new problems for students with genetic algorithms so that the games can be adapted to each players skills individually.

1 Introduction

Problem-based learning is usually considered very motivating, but we don’t have any guarantees that the learning process goes on after solving the problem and getting the feedback. The critical question is: how to generate new challenging problems from the feedback? Such problems should be individually adapted to the specific learner: strengthing ones weak areas but still giving new realistic challenges to conquer.

We also wanted to support collaborate learning. Especially if the learning happens without teacher guiding by the system or other learners is crucially important. It is also known that people learn when teaching others. The human lust for competitions should neither be ignored.

For these purposes we planned a new game environment, which all the players share but which still generates individually adapted games for each players. In our system we use genetic algorithms to generate new individual problems for each players.

Our original system was meant for teaching mathematics (functions, integrating, derivating) in highschool, but now we represent a novel version, which is specialized for teaching theoretical foundations of computer science i.e. regular expressions, grammars, automata and Turing machines. Applying genetic algorithms to functions is quite straightforward, but in this new area it is little bit more complicated. We shall represent the idea of generating new regular expressions and corresponding automata. The application for context-free grammars and Turing machines will be represented in the final paper.

In this paper we shall first introduce our game environment and a few example games. After that we shall consider the idea, how to generate new problems with genetic algorithms. Finally we shall evaluate the possibilities of our system in conclusions.

2 The requirements for the gaming environment

In this chapter we present some criteria for the adaptive gaming environment providing collaborative learning possibilities from a set of games which are individually adapted to each user. Angelides and Paul [2] have explained the elements of gaming-simulation in detail but we are not going to consider our requirements as deeply. However, the basic requirements are the same as Angelides and Paul's.
Firstly we discuss how we can prevent students getting lost, and after that we give some ideas about how to promote collaborative learning in our learning environment.

2.1 How to help students to find the right path?

If the whole collection of games were available from the beginning, many students could get lost. In other words, they would not know which game they should choose or how to proceed in this new environment.

One solution is to define several stereotypes or levels of difficulty such as beginner, intermediate and expert player. A student who wants to play in beginner level will be able to choose only two or three games (the easiest ones), while expert players will have the whole collection available. Full explanation of the new games is given in beginners' level as well in order to introduce the idea and rules of the games.

When a player collects a concrete amount of points, or she gives some evidence of mastering some ability or skill (i.e. she solves difficult problems correctly), she will pass to the next level. This means that at this moment, a few new games will be available for her. On the other hand, if she selects a level which is too difficult for her, then she will be carried to a lower one. This fact might be detected, for example, when the player loses all her bonus points because she has bought lots of hints with them.

2.2 How to get collaborative learning from a set of individually adapted games?

At the very first stages of every game, the student will play alone. This will occur until she acquires some ability level. Depending on which game it is, these stages may be presented as a normal game, but also as tutorial games in which lots of hints (for example, push Enter if this is your choice) will be provided to ensure the students learn how to play.

After this training stage is finished, the student will be able to play against or in collaboration with other students. In some games several students could play at the same time in a competitive way. The aim of the game in this case would be to win the others. In other games, collaborative learning could be reached by playing in turns. This way, after first player gives a wrong solution (which is visible to every player) the hint given as feedback by the system could be used by player two. The main idea is that after several trials somebody will find the correct answer, but she has used some hints that were feedback to another players. These hints would have been unreachable for her if she had played the same stage alone. In some games cooperation would be possible for example asking for help from others or solving the problems together in real-time.

2.2.1 Adaptive generation of new games

Every game includes a large amount of stages. The most interesting feature is that an uncountable number of new problems can be dynamically generated. To make it possible genetic algorithms, which will be discussed in Chapter 3, are used. The main idea is that, after combining a pair of problems and making some kind of mutation, a new different one is obtained. It could happen that the generated problem is not acceptable (i.e. does not respect some restrictions), but the process can be repeated as many times as needed to get a valid new problem.

To make collaborative/competitive gaming more adaptive players with similar level could play together and only those players would be exposed at the same time. The concrete problem to play with will be dynamically generated (following the mentioned process) using recently solved problems of each player as seeds. Taking the information about these games as seeds, a new game with a bit more difficulty is generated and exposed to those players.
2.2.2 Encouraging tutoring between students

After having played a certain game it would be important that players that have succeeded in it could give some lessons or hints to those students who have not. This kind of collaborative tutoring between students can also be reached in a more general way with including highscore list in each of the games. We consider that it does not only motivate students to keep on playing (i.e. to reach the highest score) but it can also provoke some kind of collaboration between students. Concretely, if a student is unable to advance in a game, although she has visited the help examples many times, she will probably ask those players which are best situated in the highscore list how to play. This way, more proficient students can instruct those with lower abilities and probably learn more by themselves as well.

2.3 Possible starting points

2.3.1 EDUCO

As one possible gaming environment we could use EDUCO a learning environment supporting social navigation [1]. It is web-based and it can be used with any web-browser so the games can be added in the system in any form that the web-browser understands can be integrated to this system. EDUCO provides also a map where players can see each others movements and chat view where students can communicate with each other. It provides the basic features that one could need for described gaming environment.

2.3.2 Quake

As a game environment we can use Quake, which is an open system and free to be modified. There are editors available in the Internet and one can plan the world map, intelligent monsters and other features as one wishes. Quake implements also a possibility for several players to play in the same game world. The players may also interact with each other like in IRC. Quake offers all the properties required for adapting collaboration game environment, but it is also facinating for students, who enjoy playing the original Quake.

3 Description of example games

1. Feeding a monster: You meet a monster, which is in fact a finite automaton. Feed the monster with strings, which it accepts, or it attacks you! The same string can be used only once.

2. Finding pairs: In the table there are several cards, with a regular expression in each. Find the pairs, which have identic expressions! In variations of this game the pairs may consist of identic finite automata (after minimizing the pairs represent the same automaton) or a deterministic and an nondeterministic automaton (determinizing is needed).

3. A labyrinth game: You are given a key, which is a regular expression. You have to find your way out of the labyrinth as fast as possible. In the walls of labyrinth there are strings. By following the strings, which are defined by your key expression, you will find the shortest way out. This game may also be a competition between two players or with the computer.

4. Picking mushrooms: You enter a square in forest, where mushrooms grow. Some of them are eatable and give you points (more strength, money etc), but others are poisonous (you lose points). The mushrooms are marked with strings and you should invent a
Figure 1: Some sketches for example games.

regular expression, which defines only eatable mushrooms, but not the poisonous.

4 Genetic algorithms in generating new problems

The main feature of our self-motivating learning environment is to generate new problems from the feedback earlier games. The new problems should be adapted to the players current skills. The most difficult exercises should be practised more, but too difficult exercises should also be avoided. We have used the rule that after every solved problem a new harder problem will be generated, based on how many trials were needed to solve the earlier problems. For this purpose we have used genetic algorithms.

In the genetic algorithms the population of some individuals develops through simulated evolution. Each individual is represented by a finite string of symbols (genome), which usually encodes a solution in a given problem. In the iterative process of evolution a selected part of population, the parents, are crossed to form new individuals, the offspring. The crossover is performed by exchanging parts of parents' genomes (substrings). In addition the genomes may be mutated with some small probability, for example some symbols are changed randomly. From the new population the best ones are again selected for the further evolution.

The main difference in our game environment is that we are not generating new solutions but new problems. It is easiest to generate new problems in the levels of regular expressions. Crossing and mutating the transition functions of an automaton doesn't guarantee that we get another same kind of automaton (i.e. in the same level of difficulty). Instead, the difficulty of regular expressions can be saved and the regular expressions can always be translated to corresponding automata.

The goodness (fitness) of a regular expression depends on a player and is evaluated according the player's game history. The most difficult type of expressions are selected for crossover and mutation to get new similar type of problems. Usually students feel the most difficult such expressions, which consist of several alternative parts (separated with union operator) and which have closures of subexpressions (* operators, like \((ab\cup ba)^*\)). Crossing such expressions produce probably new same kind of operations.

In crossing the parent expressions are divided in two parts in some random point, still considering the parenthesis (no missing parenthesis are allowed). The parts are combined and new expressions are mutated. In mutation a symbol can be changed (e.g. \(a\) is changed to \(b\) or \(\epsilon\), a union operation can be added or deleted or the exponent can be changed (0, 1 or * i.e. the base is missing, present as such or given *).
Usually the crossover between parents is performed with some probability ("crossover rate"), but in our case the last two expressions of the same difficulty level may be always crossed. Also the mutation usually takes place only with very small probability, but in our game large variation is desirable and mutation can be applied to several parts of the expression with some quite high probability.

Example

Let’s suppose the parent expressions are \( r_1 = a \ast b \cup b \ast a \) and \( r_2 = ba \ast b \cup aa \ast b \). They are divided into parts \( a \ast, b \cup b \ast a, ba \ast \) and \( b \cup aa \ast b \). After crossing we get new expressions \( r_3 = a \ast b \cup aa \ast b \) and \( r_4 = ba \ast b \cup b \ast a \). The new expressions are mutated in two random points and we get \( r'_3 = a \ast bab \) (\( \cup \) is deleted and the exponent \( \ast \) is changed to \( 0 \)) and \( r'_4 = b \cup a \ast b \cup b \ast a \ast \) (\( \cup \) is added and exponent 1 is changed to \( \ast \)).

5 Conclusions

Games have been used for long time in teaching, but primarily for fun. Gaming and structures of games can be used in many ways, also applied to teaching and learning of theoretical knowledge in any kind of subjects in all levels of studying. The subject defines the way, in which it is most optimal to construct new knowledge structures. It does not need to be funny to learn, but for reaching the best quality of learning it should feel like an experience. One has challenges, a little bit of competition to maintain one’s self-motivation, one benefits from it, one gets feedback from one’s actions, one notices that one can help other students etc. All these things are also elements of games.

Adaptive systems give every student their own learning experiences, everybody feels like individual not one of the mass and everybody cooperates with each other. If there is not enough teaching capacity, the adaptive systems can give feedback like humans: individual and just suitable for students achievements. For that purpose the system must collect accurate student’s learning history and analyze it in an appropriate way.

This kind of learning situation gives a lot of research possibilities for the learning subject itself, constructing knowledge structures in subject, user modelling, artificial intelligence and educational sciences.

References
