Using Information Technology for Personalizing The Computer Science Teaching

André Prisco, Rafael dos Santos and Silvia Botelho
Center of Computational Sciences (C3)
Federal University of Rio Grande (FURG)
Neilor Tonin
Integrated Regional University
Erechim Campus
Jean Bez
Federal University of Rio Grande do Sul (UFRGS)

***Abstract*—Recommendation systems use computational techniques to select items in a personalized way to users, taking into account criteria such as history and interest. However, several authors point out that the process of recommendation in education requires models beyond the user's taste, in order to catalyze students' learning. In addition, feedback involves the student's experience. In this work we present a recommendation system of learning objects supported by a cognitive pedagogical model. The central idea of the system is to find an object that adequately challenges the student without bothering with similar problems or becoming discouraged when faced with problems beyond his or her ability. We integrate learning models into game models to integrate them into learning models. We used as a case study a virtual learning environment which has a repository with programming problems. The results indicate that, in general, when students choose more appropriate problems (ELOs similar to theirs), they get a greater number of correct answers in their submissions. When the student choose problems that do not seem to be challenging, in general, they make wrong submissions or give up learning on the platform.**

***Keywords*—recommendation systems, ELO rating, learning models, teaching programming**

I. Introduction

The problem of choosing among many objects the most appropriate for a user is a well known problem in recommendation systems. Recommendation systems use computational techniques to select items in a personalized way to users, taking into account criteria such as history and interest of the user. These systems have been built in order to recommend movies, products or sites matched with a profile. However, several authors point out that the recommendation process in education needs a different approach. While the choice of films and websites can be based on the users taste and their history of use, education requires specific models and a different analysis of these criteria, so that the recommended learning objects can improve the learning potential.

The objective of this work is to present a recommendation system of learning objects supported by a cognitive pedagogical model. More specifically, we propose to use Piaget's learning model to develop a recommendation system that enhances equilibration and cognitive conflict, bringing learning motivation and making it more effective. The central idea of the system is to find an object that adequately challenges the student without bothering with similar problems or becoming discouraged when faced with problems beyond his or her ability.

To model this system, we use the models applied in games to integrate them into learning models. Match make systems in electronic games have the objective of find the best opponents for a game. For a player who wants to improve his performance (effect his learning in the game), the best matches are those in which he can improve his technique. Weaker opponents are easier to win, but offer little learning opportunity, while too much better opponents can be demotivating to the player, who does not have the support to appropriate much more advanced techniques. So, systems try to find opponents that potentiate learning. We have adapted one of these techniques, the ELO, to recommend, rather than adversary, programming challenges for computer science students. We use as a case study a virtual learning environment that has a repository with programming problems. This platform works through a submission system, where students submit algorithmic solutions to the problems presented, receiving automatic feedback. The system is mainly used for training marathons programming, but is also applied in the classroom. With thousands of students and millions of registered submissions, the platform offers the opportunity to compare various cases and behaviors, through the history of each student on the platform.

II. Piaget's Cognitive Theory

In this section we present, under the cognitive approach and more specifically the genetic epistemology of Piaget [1], [2], [3], some concepts that we use in this work.

The cognitive approach presents the the knowledge builds man, both individually and collectively. Therefore, a society not only has knowledge but its own structure is influenced by the knowledge that it possesses. Individuals, in a similar way, have their mind structured by knowledge as a form of interaction/ adaptation to the environment and objects (concrete and abstract) with which they interact [2], [4].

Piaget focuses much of his study on child and adolescent development, ranging from the sensorimotor stage to the formal stage. The focus of our study is in the post-formal period, which is the period studying adult learning [5], [6]. We present some concepts:

978-1-5090-5920-1/17/$31.00 ©2017 IEEE

- Schema: Human knowledge, in the cognitive approach, is not simply a set of stored information. Knowledge are mind structures that relate to other structures in a complex connectionist system. Such structures are called schemas. Some more modern approaches, such as those based on neural networks, symbolic models and neuroscience, corroborate this approach [2]. Schemas are responsible for processing information, searching for patterns, making decisions, and so on.

- Assimilation: Learning takes place through a process of adaptation and this is made possible by the processes of assimilation and accommodation. When the individual finds a problem or needs to interact with the environment, he deals with the environment in order to find the best settings. The interaction can be reading a text, driving a new car model or understanding a different programming problem. When the mind finds in its schemes a way of solving or interacting, we say that the mind has assimilated the problem.

- Acomodation: When a problem can not be assimilated by the mind, the individual has two alternatives: give up or create new schemes that fill the gaps. This construction and reconstruction of schemes to assimilate new problems is called accommodation. Accommodation is the structuring process of the mind in adapting to the environment.

- Equilibration: The tendency of the human mind to respect a balance between assimilation and accommodation is called equilibration. The individual is satisfied not only by assimilating the same problems but also tends to be discouraged when faced with problems that need a more drastic accommodation to the schemes already constructed.

- Active Experience: This concept, derived from previous ones, emphasizes the fact that the mind learns by manipulating with the object. It should not be confused with necessarily concrete manipulation, but manipulated in trying to solve it. For example, we learn more effectively the problems we solve than those we read about the finished solutions.

- Cognitive Conflict: When the mind is faced with a series of problems presented in order to respect its equilibration, it can put itself in a state of cognitive motivation in learning, that is, the curiosity itself or the annoyance by not assimilating the problem, it arouses the interest in find or build a solution. Such motivation may even be independent of external elements such as school assessments [7], [2], [8].

Perhaps one of the most challenging tasks is to use technology to gather all the information we have about each student so that we can present ways to enhance this state of motivation for the learning itself.

The following section presents such concepts from the perspective of introductory programming teaching.

III. Pedagogical Models in Teaching Programming

Problem solving skills are central to learning computer science. A teaching model based only on passing information on computing is less effective than a model that includes the manipulation of knowledge through programming problems [9],[10].

However, the simple disorganized delivery of problems to be solved implies a non-student-centered approach, in which the student undergoes a repetition process applying several times the same concepts and structures.The scripts are designed for an entire classroom, not considering the individualities of each student.This situation generates demotivation and makes it difficult to follow the teaching process. For example, a class that is studying certain computing content may have students who, despite knowing the content, do not have programming skills to solve problems with such concepts.

A student-centered approach involves choosing problems that motivate the student and catalyze the learning process. For this, we have used cognitive theories, more specifically genetic epistemology, as the basis for our model of problem choice.

The practice of exercises plays an important role because it allows the student to manipulate the concepts to be learned through the challenges of programming. However, simply practicing any exercise on the subject does not guarantee such learning.

Using the equilibration theory [1], we understand that the exercise is effective when it proposes a challenge, that is, when it causes gaps in the schemes previously constructed by the student. When faced with such gaps, the student creates new structures accommodating what has been learned in the challenge to previous schemes. As discussed in [10], the new structures may reflect new concepts that need previous concepts as well as generalizations of concepts already worked on in some aspect. We then see a convergence of the approach of such authors. Other cognitive authors also corroborate with this approach, such as [11], [12].

The continuous process of challenge and overcoming is a motivator that does not depend on external reward for the student, such as assessments or punishments. The sense of mastery over the object to be learned is the catalyst for such motivation [2], [13], [14]. On the other hand, the student's interaction with exercises that do not challenge him can be more detrimental than beneficial. These do not promote learning and discourage because of the energy that the student spends on them. Exercises that are more difficult than their ability means that the student can not create new structures because they do not have enough previous schemes to assimilate the new challenge. [8], [1], [2].

Therefore, a computational model that perceives the student in their learning and thus recommends exercises as mediators in their learning process within their abilities and respecting the rhythm of their growth is a tool for the personalization and improvement of the learning of the students of computation, especially those that do not fit the current models.

IV. Recommendation System, Matchmaking and Cognitive Theory

The problem of choosing, among several objects, the most suitable for a user is a problem already worked on the recommendation systems. Recommendation systems have been built in order to recommend us movies, products or sites more suitable to our intent. However, the process of recommendation in education does not follow the same rules, as stated out by [15].

In the choice of movies we can base on the intentions of the user, while in education we should not always recommend to the student what he or she wants, but the most appropriate learning object, that is, the one that enhances their learning. Furthermore, feedback is not simply the acceptance or use of the object, but an evaluation of the entire student experience with it.

According to [15], the specific problems for a recommendation system for education are:

- Items chosen by students may not be pedagogically appropriate for them;
- Personalization should not be done only on the choice of learning objects, but also on the feedback of their uses;
- Students do not want to read many articles.

In [16] are presented some learning environments that implement recommendation systems. [17], [15] and [18] propose web systems that indicate the best navigations of the users, so that they find the most suitable learning objects, mainly taking into account their feedbakcs and browsing histories. Another work that aims to adapt the presentation and navigation system of online courses, based on the level of knowledge of a student, is presented in [19].

A tutoring system to help students learn programming is presented in [16]. Protus was designed and implemented as a mentoring system, able to recommend useful and interesting materials to students based on their different knowledge, preferences, learning purposes and other characteristics. The system recognizes the learner's learning style and allows him to follow different paths in the development of learning activities. The AprioriAll pattern mining algorithm [20] is adopted to extract behavioral patterns from the log of activities done by students and recommend the most appropriate learning paths.

Many recommendation systems for education need metrics to classify learning objects and students, so that relationships can be established between students and the activities most appropriate for them. In the work [21], an adaptive classification mechanism for personalization of teaching using repositories on the Internet is presented. The system uses approaches based on the preferences and interests of neighbors to classify the degree of relevance of learning objects according to the intention of a user.

With respect to these metrics, the theories developed for games can be useful. Matchmaking is the task that game producers and developers make to choose the best opponent for a player [22]. In this case, the goal is not to choose the opponent that most pleases, but the one that enhances the skill gain and minimizes the risk of demotivation. In the case of the method created by [22], players are chosen for a match in order to have a greater gameplay, taking into account factors such as language, skill level and location.

One of the most used models for this task was developed for chess tournaments and today is used in various sports and electronic games. ELO [23] aims to quantify players through their game history. Elo is a statistical ranking system that can calculate relative skill level values for competitors or machines in competitive games. The ranking was created by an american physicist, born in Hungary, called Arpad Elo. The ELO system works as follows:

- Each player has an skill value (ELO);
- At each match, the ELO values of the players involved are updated;
- Before the match occurs, there is a calculated probability (based on the ELOs of each players) of player 1 to beat player 2 (and vice versa);
- After the game, if the win occurs by player with most likely to win, the ELOs of both players practically do not change;
- After the game, if the win occurs by player with less likely to win, the ELOs of both are significantly changed, the winner having his ELO increased and the loser decreased;

The proof of ELO probabilities and update calculations can be found in [24].

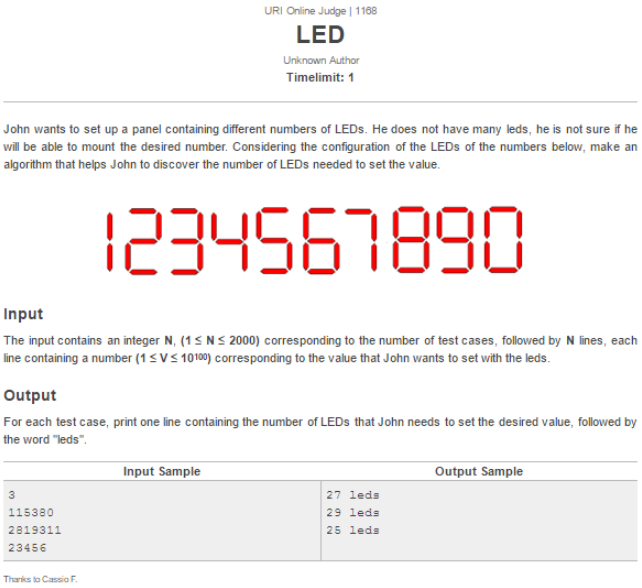
For this work, we present the hypothesis that the process of improving in a game is an equilibration process, presented in the previous chapter. In choosing the right players to tackle (matchmaking), ELO arises as a facilitating mechanism for the equilibration process. Players have a scheme formed to deal with the game. When faced with difficult players and with adequate ELOs, they do not have formed schemes and need to go through the stages of assimilation and accommodation, so that they acquire new skills (increasing their ELOs), finally reaching a state of equilibrium. In these situations, we believe that subjects are motivated and learn through the manipulation of appropriate objects (matches with a certain degree of challenge) and of the cognitive conflict provoked.

Thus, this type of game can be treated as a learning process of Piaget and the performance metrics of games can be an indicator for the evaluation of learning.

V. Online Judges Platforms as Virtual Learning Environment

Online Judges are websites designed to support events such as the ACM International Collegiate Programming Contest (ICPC). In these events, computer students form teams to compete with each other to solve programming problems. The competition is well-known and students from all over the world participate in it.

Online Judges tools are virtual learning environments that have a repository of learning objects. In these VLEs, students



URI Online Judge | 1168

LED

Unknown Author

Timelimit: 1

John wants to set up a panel containing different numbers of LEDs. He does not have many leds, he is not sure if he will be able to mount the desired number. Considering the configuration of the LEDs of the numbers below, make an algorithm that helps John to discover the number of LEDs needed to set the value.

1234567890

Input

The input contains an integer **N**, ($1 \leq N \leq 2000$) corresponding to the number of test cases, followed by **N** lines, each line containing a number ($1 \leq V \leq 10^{100}$) corresponding to the value that John wants to set with the leds.

Output

For each test case, print one line containing the number of LEDs that John needs to set the desired value, followed by the word "leds".

Input Sample	Output Sample
3 115380 2819311 23456	27 leds 29 leds 25 leds

Thanks to Cassio F.

Fig. 1. Example of problem in a platform

find, in addition to the learning objects, a series of tools that support their learning: discussion forums, classification of learning object by level of difficulty and subject, monitoring of their evolution, among others.

The learning objects provided in these VLE have a specific format:

- Problem name;
- Problem number;
- Maximum execution time;
- Difficulty level of the problem;
- Category of the problem;
- Texts and explanatory images about the problem to be solved;
- An explanation and an example of how data entry should be;
- An explanation and an example of how data output should be.

Figure 1 presents an example of a learning object on this platform.

In addition, the manipulation on these learning objects are also particular:

- Students should make a program in a programming language to solve the problem proposed in the learning object;
- After, the student must submit his solution to the platform to be evaluated;
- The platform has an automatic judge that analyzes the answers of the program submitted by the student and gives a feedback;
- The feedback can be:
  - – Accepted: problem was accepted without any error;
  - – Closed: there was a problem connecting to the server and the submission was not received;
  - – Compilation error: some structure of the code was written in the wrong way;
  - – Possible runtime error: possible error in the execution flow of the program;
  - – Presentation error: the output data is not formatted as described in the problem statement;
  - – Runtime error: error in program execution flow;
  - – Time limit exceeded: execution time exceeded the stipulated in the problem statement;
  - – Wrong answer: program performed normally, however, it presented incorrect outputs.

The URI Online Judge is the main tool of this type in Brazil. Developed at the Integrated Regional University - Erechim campus, the system is used in contests of several universities in the country. In addition, URI Online is available so that any student can practice programming problems and thus, teachers and students have begun to incorporate the use of the URI Online system no longer for competitions, but as a didactic tool. Currently it has more than 1500 problems, thousands of users and more than 4.5 million submissions.

VI. Metodology

In this section we present the methodology used to process and analyze the URI Online Judge data. As we presented in the previous section, the main users are students who submit their solutions to programming problems as an activity evaluated in their courses, students who want to practice independently and students who train for programming contests. These are free to choose the problems they want to solve. In our methodology we did not choose to insert the recommendation system into the platform and then observe the results. We chose to observe the platform history, in which the students freely chose the problems, and we observe those who followed or did not follow the guidelines that we would give if the system had recommended.

We consider the frequency of use of the tool, non-abandonment and effectiveness in solving problems as indicators of learning. In this way, we observe the behavior of the ”successful” students and verify how naturally they followed our cognitive model. We also observed that most of the students who followed our model were successful. These two measures are the first indicators that will inform if the model is appropriate.

Note that despite the large volume of data, we have little concrete information about each student. For example, we do not know how long it took him to programing, whether he got help from colleagues or from a site. We do not have access to the algorithm he wrote. We have only the problem information (as showing in section V) and the submission information (as shown in figure 2 and in table I). So, we will focus on the evolution of student skills through their submissions. Other attributes such as the association between problems and the time between submissions will also be considered in the future, although they are not addressed in this paper.

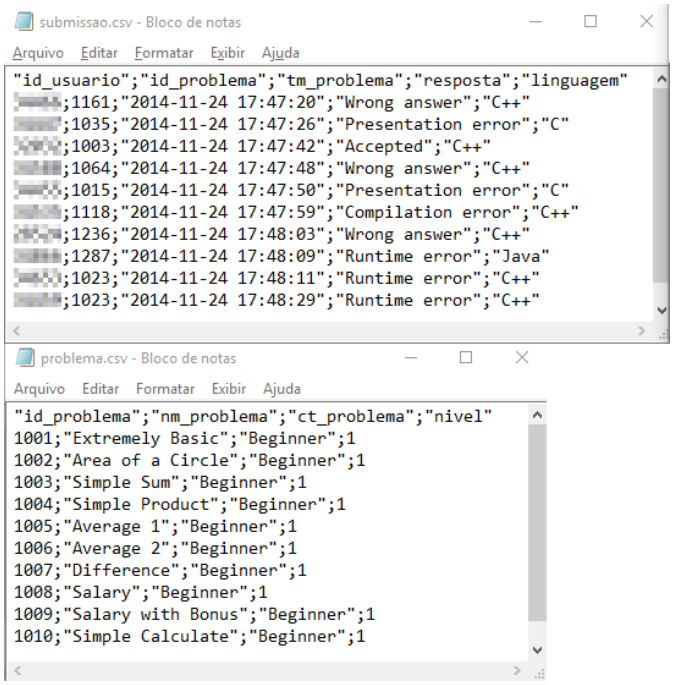


Fig. 2. Sample of the URI log files

User Id	Problem Id	Submission Time	Feedback	Language
7XXXX	1134	”2016-05-05 01:10:09”	Accepted	C
7ZZZZ	1154	”2016-01-04 20:41:39”	Accepted	Java
4XXX	1002	”2015-02-03 20:57:46”	Wrong Answer	C++

TABLE I. SUBMISSION DATABASE SAMPLE

Data analysis involves a change in the use of ELO that we discussed in the introduction. While ELO deals with a player's relationship to another player, our approach adapts the metric to relate the student to the problem being attempted. Genetic epistemology is relational, that is, it considers learning as a relation of the individual to the object to be learned. In this way, assigning ELOs to problems as well as students and then comparing them is an elegant way to modify the metrics to fit our model.

We regard each submission as a game, a ”duel” between the player and the problem. As discussed by Piaget, learning allows to modify and manipulate the object to be learned, to the same extent that modifies the individual himself. Such a concept is modeled from the variation of ELOs of student and problems. At each beginning of submission, our algorithm has stored the ELO of the problem and student. A positive feedback (accepted) indicates that the student ”won the game” while a negative feedback (wrong answer, timelimit and etc) indicates that the ”problem has won.” After the evaluation, the ELOs are then updated.

Each problem is a learning object that does not have its content modified over time, so its ELO variation must be controlled. Here's how we approach this problem.

The methodology adopted follows the figure 3. The data collected from the URI came as in the figure 2 and are stored in a DBMS (we use PostgreSQL [1]). In the database, we

[1]https://www.postgresql.org

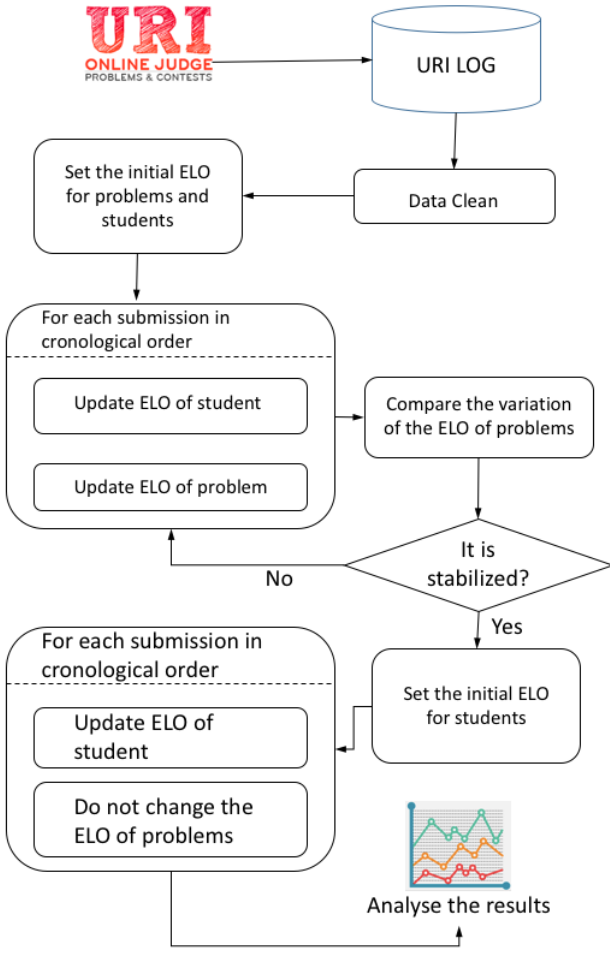


Fig. 3. Metodology adopted for URI log analysis

Level	Elo
1	1100
2	1200
3	1300
4	1400
5	1500
6	1600
7	1700
8	1800
9	1900
10	2000

TABLE II. INITIAL ELO ACCORDING TO THE LEVEL OF THE PROBLEM

did simple treatments, such as eliminating replicated data and transforming textual data into numerical data (to facilitate processing). Next, we assign an initial ELO for each student and for each problem. At this initial stage, all students receive an mean ELO (1100). For the ELO assignment of the problems, we use as the heuristic the level of difficulty indicated by the problem author (table II. There is no need to accurately assign an initial ELO since the iterative algorithm will fine-tune. However, a good initial value provides a faster convergence, which facilitates the experiment.

With the initial ELOs assigned, our algorithm simulates each submission performed in chronological order. Through feedback from each submission, ELOs are updated. At the end of the process you have the final ELOs of each student

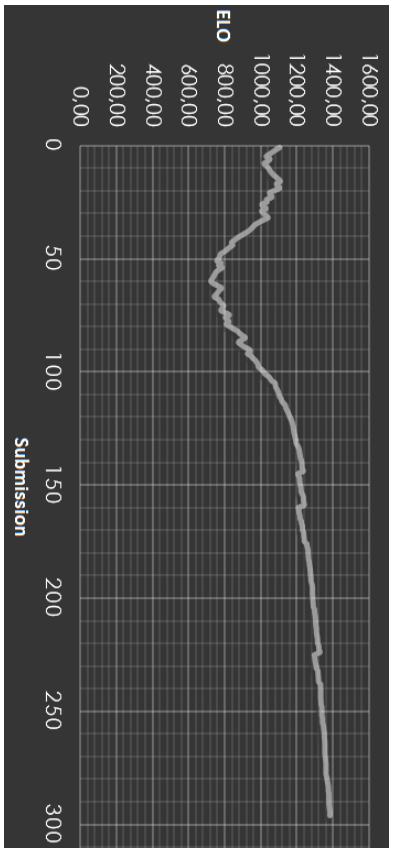


Fig. 4. History of a Student's ELO

and each problem. The values at the end of the process are then compared with the pre-process values. If the difference between the values is substantial this indicates that the ELOs have not yet converged and the process is repeated. As the process converges, we store the final ELOs of each problem and we consider it constant. This is another modification we did in the ELO metric. Since each problem is a learning object that does not have its content modified over time, the ELO variation must be restricted to the convergence step. Once we know the ELO of each problem, we consider it to be constant for the next step.

Once the convergence step has been completed, students again receive mean ELOs and for each new submission their ELOs are updated and stored in the database. We then have the history of the ELO of each student for each submission, and analyzes can be made as in the figure 4. With these data, the analyzes we present in the VII section can be performed.

The following section presents the results obtained and the conclusions.

VII. Results and Conclusion

After applying the methodology, we analyzed the results separately in 2 groups. In the first group, all the submissions were observed in which the student's ability seemed to be adequate for the chosen problem, that is, problems with ELO similar to that of the student. Already for the second group of analysis, all submissions were observed in which the student's ability appeared to be inadequate for the chosen problem, that is, problems with ELO significantly different from that of the student.

Analysis Group	Number of submissions	Accepted	Error
Suitable ELOs	1 million	54%	46%
Inappropriate ELOs	3.5 millions	41%	59%

TABLE III. Analysis of learning in submissions

The table III shows the results of these analyzes.

It was first verified that, of the 4.5 million submissions in the database, about 1 million were classified as adequate submissions, that is, the student chose a learning object that seems to be ideal, with ELO similar to theirs. On the other hand, 3.5 million of the learning objects chosen by students were classified as inadequate, with ELO significantly different from their own.

Of the 1 million submissions considered adequate, 54 % got the Accepted response, ie, the students got the problem right. While from these same submissions, 46 % got some kind of error.

Regarding the second group of analysis, of the 3.5 million submissions considered inadequate, 41 % obtained the Accepted response, that is, the students got the problem right. While in these same submissions, 59 % got some kind of error.

The results indicate that, in general, when students choose more appropriate problems (ELOs similar to theirs), they get a greater number of correct answers in their submissions. Already when the choice is made by problems that do not seem to be challenging or demotivating, in general, they err more or end up giving up learning on the platform.

Considering all submissions with positive feedback, the ELO of the problem is on average 5 points above the ELO of the student. For submissions with negative feedback, this averaged over to 132, indicating that positive feedbacks submissions occur in closer ELOs.

Finally, it should be noted that the observation and analysis of data in education (Learning Analytics) is a vast area and very promising. Using the area of recommendation systems can greatly help the understanding of the profile of the student and personalize the teaching of programming, respecting the peculiarities of each student. The analyzes made for this work represent the first steps in this direction. There is much to be done, considering other information not explored. Other metrics such as TrueSkill [25] are being implemented and we want to present our results in a future work. We believe that new results will bring useful information to teachers and students.

References

[1] J. Piaget, "Intellectual evolution from adolescence to adulthood," *Human development*, vol. 15, no. 1, pp. 1–12, 1972.

[2] G. R. Lefrançois, *Theories of human learning: What the professor said.* Cengage Learning, 2012.

[3] F. Becker, "O que é construtivismo," *Revista de educação AEC, Brasília*, vol. 21, no. 83, pp. 7–15, 1992.

[4] A. Garnham, "Cognitivism," 2009.

[5] P. K. Arlin, "Cognitive development in adulthood: A fifth stage?" *Developmental psychology*, vol. 11, no. 5, p. 602, 1975.

[6] C. C. Knight and R. E. Sutton, "Neo-piagetian theory and research: enhancing pedagogical practice for educators of adults," *London Review of Education*, vol. 2, no. 1, pp. 47–60, 2004.

[7] S. A. da Silva, "Conflito cognitivo: Herói ou vilão?" *Schème-Revista Eletrônica de Psicologia e Epistemologia Genéticas*, vol. 4, no. 1, pp. 209–238, 2012.

[8] M. A. Moreira, *Teorias de Aprendizagem*, 2nd ed. Editora Pedagógica e Universitária, 2011.

[9] A. Pears, S. Seidman, L. Malmi, L. Mannila, E. Adams, J. Bennedsen, M. Devlin, and J. Paterson, "A survey of literature on the teaching of introductory programming," *ACM SIGCSE Bulletin*, vol. 39, no. 4, pp. 204–223, 2007.

[10] D. B. Palumbo, "Programming language/problem-solving research: A review of relevant issues," *Review of educational research*, vol. 60, no. 1, pp. 65–89, 1990.

[11] D. P. Ausubel, J. D. Novak, H. Hanesian *et al.*, "Educational psychology: A cognitive view," 1968.

[12] C. R. Rogers and H. J. Freiberg, "Freedom to learn," 1969.

[13] J. A. da SILVA, "Repetição e desafio nos exercícios escolares: dois lados de uma mesma moeda," *Schème-Revista Eletrônica de Psicologia e Epistemologia Genéticas*, vol. 1, no. 1, pp. 95–107, 2011.

[14] R. M. Ryan and E. L. Deci, "Intrinsic and extrinsic motivations: Classic definitions and new directions," *Contemporary educational psychology*, vol. 25, no. 1, pp. 54–67, 2000.

[15] T. Y. Tang and G. McCalla, "Smart recommendation for an evolving e-learning system: Architecture and experiment," *International Journal on elearning*, vol. 4, no. 1, p. 105, 2005.

[16] A. Klašnja-Milićević, B. Vesin, M. Ivanović, and Z. Budimac, "E-learning personalization based on hybrid recommendation strategy and learning style identification," *Computers & Education*, vol. 56, no. 3, pp. 885–899, 2011.

[17] A. Krištofič, "Recommender system for adaptive hypermedia applications," in *IIT. SRC 2005: Student Research Conference*, 2005, p. 229.

[18] R. Farzan and P. Brusilovsky, "Social navigation support in a course recommendation system," in *International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems*. Springer, 2006, pp. 91–100.

[19] C. Romero, S. Ventura, and P. D. Bra, "Knowledge discovery with genetic programming for providing feedback to courseware authors," *User Modeling and User-Adapted Interaction*, vol. 14, no. 5, pp. 425–464, 2004.

[20] W. T. H. Pi-lian, "Web log mining by an improved aprioriall algorithm," *Engineering and Technology,* , vol. 4, no. 2005, pp. 97–100, 2005.

[21] K. H. Tsai, T. K. Chiu, M. C. Lee, and T. I. Wang, "A learning objects recommendation model based on the preference and ontological approaches," in *Advanced Learning Technologies, 2006. Sixth International Conference on*. IEEE, 2006, pp. 36–40.

[22] J. Davis, M. Feldman, and B. Johansen, "System and method for combining automatic opponent matching for computer gaming with chat room searchers," Aug. 8 2003, uS Patent App. 10/637,048.

[23] A. E. Elo, *The rating of chessplayers, past and present*. Arco Pub., 1978.

[24] A. Elo, "New uscf rating system," *Chess Life*, vol. 16, pp. 160–161, 1961.

[25] P. Dangauthier, R. Herbrich, T. Minka, T. Graepel *et al.*, "Trueskill through time: Revisiting the history of chess." in *NIPS*, 2007, pp. 337–344.