A Recommender System of Computer Programming Exercises based on Student's Multiple Abilities and Skills Model

1st Fabiana Zaffalon
PPG Sciences Education
Federal University of Rio Grande
Rio Grande, Brazil
fabinhazaffalon@gmail.com

2nd André Prisco
PPG Sciences Education
Federal University of Rio Grande
Rio Grande, Brazil

3rd Ricardo de Souza
PPG Computational Modeling
Federal University of Rio Grande
Rio Grande, Brazil

4th Davi Teixeira
Center of Computational Sciences
Federal University of Rio Grande
Rio Grande, Brazil

5th Wanderson Paes
Center of Computational Sciences
Federal University of Rio Grande
Rio Grande, Brazil

6th Paulo Evald
Center of Computational Sciences
Rio Grande, Brazil

7th Neilor Tonin
Integrated Regional University
Erechim, Brazil

8th Sam Devincenzi
Center of Computational Sciences
Federal University of Rio Grande
Rio Grande, Brazil

9th Silvia Botelho
PPG Sciences Education
Federal University of Rio Grande
Rio Grande, Brazil

***Abstract*—This paper presents a programming exercise recommender system based on the Student's Multiple Abilities and Skills (SMAS) model, which is developed from Item Response Theory and Elo System Classification, for estimation of multiple student's abilities. This model assumes that programming exercises have many ways to be solved (paths) and each path requires different abilities from the student. To evaluate the recommender system, an experiment was conducted in a class of Algorithms and Data Structures I. For this study case, the recommender was connected to an Online Judge system that had a programming problem base. The results show that the proposed recommender has the ability to indicate relevant problems according to the student's abilities.**

***Index Terms*—Recommender system, Programming, Student's abilities, Student's skills.**

I. Introduction

With the amount of resources available in massive environments, one of the main challenges faced by educators is to select and propose personalized content and exercises to students, according to their abilities. To help educators, there are educational recommender systems, which have the purpose of indicating study materials and exercises according to the student's abilities.

In programming disciplines, selecting exercises for a great class is an even more complex challenge, because programming exercises involve multiple skills, and some problems can be solved in several ways [1]. It is highlighted that each way to solve the problem consists of a set of skills. Currently, the contributions of Computational Thinking have been discussed for problem solving, mainly in the computing area, but also in propaedeutic disciplines [2]. For programming students, there are platforms that have a repository of programming problems, called *Online Judge*, where users choose a problem, submit the solution and receive automatic feedback. For an adequate recommendation, there are models that use metrics based on expected performance to estimate student skills. The basic principle of these models is to update skills according to the expected result after the subject (student) interacts with the object (programming problem). That is, when a student with high ability solves an easy problem, there are no surprising results; therefore, their abilities will be updated with a small value. However, if the student correctly solves a hard problem, their abilities will be updated to a high value. Among models based on performance expectations, the Item Response Theory (IRT) and the Elo Classification System stand out.

The IRT is a set of mathematical models that seeks to represent the probability of an individual correctly answering an item, as a function of the item's parameters and the student's ability [3]. The IRT presents multidimensional models that consider students to have multiple abilities [4]. On the other hand, the Elo model, used for evaluation in international chess rankings, aims to classify competitors through their game histories, using a statistical classification that calculates their skills, relating a player to another player [5]. In the application of Elo in education, the student is a player, while the problem is the opponent. In this way, it updates the student's ability and the difficulty of the problems with each interaction between

This study was funded by the Coordination for the Improvement of Higher Education Personnel - Brazil (CAPES) – Finance Code 001.

them [6].

The Student's Multiple Abilities and Skills (SMAS) model, based on performance expectations, estimates the multiple abilities of students in the context of massive online programming education, where the unique accessible information about students is the final interaction result (correct or incorrect answer). The SMAS model estimates the probability of success of each *path* of problem solution, through the Multidimensional IRT, and updates the student's abilities through the Elo model.

In the context of massive online education, the adoption of skill representation and assessment techniques, guided by methods based on performance expectations, is important for recommending learning objects that involve multiple skills [7]. Therefore, the objective of the recommender system proposed in this work is to indicate programming problems that stimulate the student's multiple abilities and challenge them, without demotivating them by facing problems beyond their capacity, or making them bored by suggestions of programming problems too easy for their current abilities and skills. The SMAS model was connected to an Online Judge system for programming problems. From the estimation of the probability of success of each *path* that can solve the problem, the recommender system indicates a problem from the Online Judge database that is in accordance with the current student's abilities, based on recommendation heuristics. This paper is organized as follows: Section II provides a theoretical background about abilities involved in programming exercises, recommender systems, and online judges. In Section III, the models based on performance expectations are discussed, followed by a presentation of SMAS, in Section IV. Finally, in Section V, a case study is discussed, and the conclusions are given in Section VI.

II. Theoretical background

This section aims to provide a theoretical background about skills involved in designing programming exercises, recommender systems, and online judges.

A. Skills involved in designing programming exercises

Teaching programming enables students to develop computer systems that are able to solve real-world problems [8], [9]. However, as programming is context-dependent, its learning is a complex task, which requires multiple skills [10]. There is a skill set involved that goes beyond knowing the programming language syntax [9]. They are: mathematical ability, logical reasoning, text interpretation, among others [1].

Computational Thinking, CP, is the process of formulating and solving a problem using the fundamental concepts of computer science. In the most diverse areas, these concepts can be applied to help in solving real-world problems [11]. In regard to computer science, some elements of CP can be identified in the source-code of the algorithms [2], [12], such as:

- Data analysis: write a program to solve basic statistical calculations [13]. In source-codes, this ability can be identified by the use of arithmetic operators [12].
- Data representation: use data structures to solve a problem [13]. In programming teaching, it is understood that this skill is developed through exercises that use identifiers, arrays, pointers or data structures [12].
- Abstraction: encapsulate a set of commands that are repeated with the same purpose, as well as the use of conditionals, loops, recursion, and so on [13]. Therefore, this skill is developed through the occurrence of logical operators used in conditions, repetition commands and recursive calls [12].

B. Recommender Systems

Recommender systems are used to offer suggestions for products, services and information to the users [14]. The purpose of these systems is to assist users in the process of searching and selecting content, indicating the content that most closely matches their characteristics (or profile), through an information filtering system, where people provide some search information and the system presents a set of content that fits in this search [15]. It is highlighted that recommender systems built for education are important resources, because they provide easy access to study materials and help students to find and select content focused on their profiles [14]. They also act as an information filter, indicating the material that best meets the student's needs [15].

For recommendation systems that propose exercise solutions, a model to estimate strengths and weaknesses in the student's skill set can provide a more adequate indication of exercises. It avoids frustration, generated by problems that are too difficult; or demotivation, caused by problems that are too easy for the current skills and abilities of a student [7]. Therefore, a first step towards these systems is to perceive and represent metrics that make it possible to accurately trace the evolution of competence and skills of learners.

C. Online Judge

Online Judges are platforms aimed at evaluating algorithm source-codes in the most diverse programming languages. The objective of these platforms is ensuring a reliable and continuous assessment, based on algorithms that are submitted by users, which are distributed worldwide. Many universities offer this type of system to help their students prepare for competitive programming championships [16]. An advantage of using these platforms is the immediate feedback, allowing students to correct their mistakes and resubmit their solutions for an automatic re-assessment [17], [18].

III. Models based on performance expectation

Expected performance calculations are computational and statistical techniques that infer the probability of a given subject having a successful interaction with the learning object. Most of the techniques are applied in games to predict the results or to create mechanisms for ranking competitors [19].

In education, there are models that use rating metrics based on expected performance to estimate student's abilities. The basic principle is to update skills according to the expected result after the subject's interaction with the object: if a student with high ability correctly solves a problem considered easy, the results will not be surprising and the update will be small; otherwise, the update will be higher [5]. Among these models, the Item Response Theory [3] and the Elo Classification System [20] stand out.

A. Item Response Theory

Item Response Theory, IRT, allows a precise analysis of each item that makes up the assessment instrument, taking into account the item's characteristics in the skill production process [3], [21]. To calculate the skill estimate, θ, the IRT is based on statistical methods and mathematical models that consider individual responses and item properties [3], [21]. Naturally, the greater the skill of an individual, the greater will be the probability of getting an item right [3].

The IRT has models for unidimensional tests, which imply the existence or predominance of only one skill; and multidimensional models, which are applied to assess more than one skill. Research has shown that Multidimensional Item Response Theory (MIRT) model adapts better to real data than unidimensional models, because, in education, subjects' responses are determined by more than one skill at the same time [22]. Furthermore, MIRT models can be compensatory or non-compensatory. A compensatory model, shown in (1), considers that the item's solution probability is maintained or increased even if one of the skills is low, which is compensated by a higher skill [23].

$$P_{ij} = P(Y_{ij} = 1) = \frac{e^{(\sum_{m=1}^{M} \alpha_{jm}(\theta_{im} - \beta_j))}}{1 + e^{(\sum_{m=1}^{M} \alpha_{jm}(\theta_{im} - \beta_j))'}}, \quad (1)$$

where Y_{ij} is the individual response i to the item j; $P(Y_{ij} = 1)$ is the probability of correct answer; α_{jm} is the item discrimination parameter j in the dimension m; θ_{im} is the ability of the individual i in the dimension m; and, β_j is a scalar that indicates the item's difficulty j.

On the other hand, the non-compensatory MIRT model, shown in (2), assumes that the dimensions of knowledge are independent of each other, not being possible to compensate one skill with another. Therefore, it is assumed that the student has to possess all relevant skills to answer an item in the correct way. Thus, in MIRT model, each dimension's logistic function is an independent probability function [24].

$$P_{ij} = P(Y_{ij} = 1) = \prod_{m=1}^{M} \frac{e^{(\alpha_{jm}(\theta_{im} - \beta_j))}}{1 + e^{(\alpha_{jm}(\theta_{im} - \beta_j))}}. \quad (2)$$

B. Elo Rating System

The Elo classification system is a well-known model, which was proposed to classify chess player performance [20]. Through statistical methods, each player receives an initial rating, θ_i; and, as they participate in the games, their ratings are updated according to the obtained results. As aforementioned, when the Elo system is used in education, it considers that a student is a player and the opponent is the problem to be solved [5]. Therefore, at the end of each problem resolution, the player classification is updated [5].

Elo works continuously, according to results and expectations. The expected probability that the player will win the match is given by the logistic function with respect to the difference in the estimated ratings [5],

$$P(R_{ij} = 1) = \frac{1}{1 + 10^{\frac{\theta_j - \theta_i}{400}}}, \quad (3)$$

where $R = \{0,1\}$ is the result set of a game: 1 (win) and 0 (lose). Given a game between player i and player j, with Elo θ_i and θ_j, respectively, at the end of a game, new Elos are calculated according to the result expectations, the previous Elos and a constant k. The higher the k, the greater the Elo change [5],

$$\theta_i = \theta_i + k(R_{ij} - P(R_{ij} = 1)). \quad (4)$$

The use of the Elo classification system has some interesting characteristics for its use in education, such as: simplicity of use in online environments and easy implementation in educational systems, due to the low number of parameters to adjust [5]. However, as a drawback, it is intended to track only a single skill [25].

C. Student's Multiple Abilities and Skills Model

The Student's Multiple Abilities and Skills model, SMAS, is also based on performance expectations. Its main objective is to estimate, simultaneously, the multiple abilities of students who use massive online systems with automatic correction. This model assumes the following assumption: programming problems have more than one way to be solved, called *paths*, and each of the *paths* has a set of skills needed to solve it correctly. For each *path* it is necessary that the student has all the skills necessary to solve the problem; that is, the skills are not compensatory. Among the possible *paths* of solution, it is understood that they are mutually compensatory, as a student can solve a problem, even with some lower skills, if the *path* chosen does not require such skills. Therefore, each problem j has a set of *paths* to the solution, $j = \{s_1, s_2, .., s_n\}$, each set s is composed of n skills that have difficulty β and relevance α, $s = \{(\beta_1, \alpha_1), (\beta_2, \alpha_2), ..., (\beta_n, \alpha_n)\}$. Relevance is how important a given skill is for the correct resolution of the problem, described by a real value between 0 and 1. Therefore, a relevance of 0 means that the specific skill is not needed.

As all skills are needed on each *path* s; then, for each *path*, the probability of success is calculated, using the non-compensatory MIRT, as follows

$$P_{is} = \prod_{n \in s}^{N} \frac{e^{(\alpha_{ns}(\theta_{in} - \beta_{ns}))}}{1 + e^{(\alpha_{ns}(\theta_{in} - \beta_{ns}))'}}, \quad (5)$$

where P_{is} is the probability that the student i gets the problem right with the *path* s; α_{ns} is the relevance of skill n in *path*

s; β_{ns} is the difficulty of skill n on *path* s; and θ_{in} is the student's n skill i.

It is understood that the student will choose that *path* that presents the highest probability of being correct, because according to Rossler [26], the accomplishment of individuals' tasks, based on spontaneity, tends to follow the law of least effort, which requires less energy, time and thought. Thus, the probability of the question being correct assumes the highest probability among the *paths* ($P_{ij} = P_{is}$), as well as the problem parameters, such as: difficulties and relevance of *path* skills with higher probability.

After each answer, the problem difficulty and student's skills are updated using the adapted Elo model [27], as shown in (6), which uses the difference between the observed performance, Y_{ij}, and the expected performance, P_{ij}.

$$\begin{aligned} \theta_{in(t)} &= \theta_{in(t-1)} + \alpha_{jn} K\{Y_{ij} - P_{ij}\}, \\ \beta_{jn(t)} &= \beta_{jn(t-1)} - \alpha_{jn} K\{Y_{ij} - P_{ij}\}, \end{aligned} \tag{6}$$

where β_{jn} and α_{jn} are the difficulty and relevance of skill n of problem j, respectively, both from the *path* with the highest probability of success; K is a constant equal to 32[1], which defines how much the estimate can be affected by the difference between the current and the expected response; Y_{ij} is student i's answer to problem j, being 1 for correct answer or 0 for error.

IV. SMAS Model-based Recommender System

The SMAS model was incorporated into a recommender system, named Primary [19]. Initially, the Primary was developed to recommend math exercises for students entering university, coursing pre-calculus [19]. This system used the Elo model to estimate and update student's abilities and problem difficulty.

In this work, the Primary has been adapted to recommend multi-skill programming exercises. Therefore, the Elo model was replaced by the SMAS model. Furthermore, the Primary was connected to the *beecrowd*[2] Online Judge, which contains a database of programming problems available for students or users who want to solve exercises or train for programming competitions. Thereby, the Primary, tied to the *beecrowd*, allows registered students to receive recommendations for problems hosted on the platform, according to their abilities.

As the problems registered in *beecrowd* have not identified the skills involved for their solutions, an expert analyzed a set of 200 problems from this repository. Therefore, all possible *paths* of problem solution were identified. In each *path*, the required skills were highlighted for problem solution, and each skill received a difficulty value and a relevance value, which belongs to an interval between 0 and 1. Relevance equal to 1 indicates that the skill is indispensable for a positive solution, while relevance equal to 0 indicates that the skill is not needed to solve this specific exercise [27].

[1]empirically adopted value, proposed in [5]

[2]https://www.beecrowd.com.br

A. Recommender system architecture

Figure 1 shows the recommender system architecture, which has three modules:

1) Representation module: stores data about students and learning objects;
2) Mediation module: uses information from the representation module for decision making;
3) Evaluation module: updates the representations.

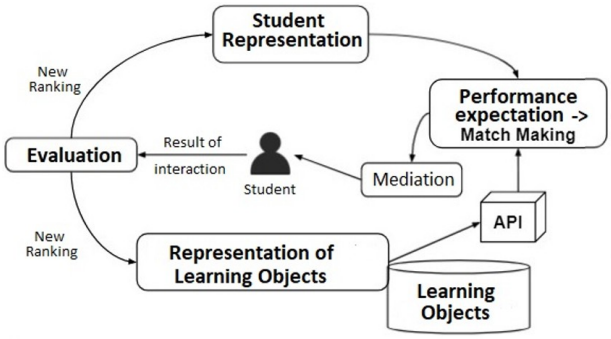

Fig. 1. Architecture of the recommender system, adapted from [19]

The *Student Representation* module stores student data, such as: identification (ID) and skill values, while the *Representation of Learning Objects* module stores the object identification, the difficulty level and the relevance of each skill required by the object.

It is highlighted that objects are stored in the *beecrowd* platform database. Therefore, an API for integrating the recommender system with the platform was also implemented, in Python language. The recommender system indicates the problem to be solved and directs the student, through a link, to the *beecrowd* site, where the student interacts with the object. After submission, the API collects, in real time, the result referring to the interaction and stores it in the *Evaluation* module.

The evaluation takes place through the result of the subject's interaction with the learning object. This interaction is analogous to a duel between subject and object, because the problems challenge the student. If the interaction's result is successful, that is, the student gets the right solution to the problem, it is said that the student won the duel. Otherwise, in which the student misses the solution, it is said that the object won. Thus, both student skills and problem difficulties are updated after the result of each interaction between a student and a problem.

A value is assigned to the student's initial skill. When students get the solution right, their skills involved in solving the problem increase and another problem is indicated. In case of a wrong solution, their skills decrease, and the problem goes back to the problem suggestion database, to be recommended in the future. Next, the recommender suggests other problems for the student. If the student does not want to solve the proposed problem, there is the option *Skip* (do not solve), in

which case, their skills are also reduced and a new problem will be recommended.

The *Mediation* module selects the learning objects, in a personalized way, according to the student's abilities and problems, through the *Representation* module. Some issues are pre-selected and stored in the recommendation zone. For each problem, the probability of the student getting the solution right is estimated, considering all *paths* of the solution. All problems with a probability of success between 40% and 60% are part of the recommendation zone and can be indicated. There are 4 heuristics for recommending learning objects that are part of the recommendation zone. They are:

1) *Minimum*: work and develop student's lower abilities. A preliminary analysis of the student's abilities is made and the lowest is identified. Among the problems in the recommendation zone, the one that has greater relevance in the low ability of the student is selected;
2) *Maximum*: work the highest student's abilities. After analyzing the student's skills, the highest ones are identified and a problem is recommended, which belongs to the recommendation zone, with greater relevance to the identified skills;
3) *Average*: the problem with average relevance is recommended according to the student's abilities in an equilibrated way, without pondering significantly like *Maximum* or *Minimum* heuristics;
4) *Random*: any problem that belongs to the recommendation zone is recommended.

When the recommender system finds more than one problem that meets the recommendation requirements or that has the same relevance values in the skills, the indication is made randomly among these problems. Initially, the recommender system applies the *Average* heuristic until there is a discrepancy of 20% between the values of the student's abilities. As the system works with a cold start, that is, all skills have the same initial value, then, the first submissions serve as a gauge of student's skills, being used to differentiate the skill values. From real ability values, the system applies other heuristics.

At each submission, the skills are analyzed and, after the calibration, the system adopts the *Minimum* heuristic in order to work with the low skills of the students. There is a control in relation to the result of each submission; so, if the student submits more than three consecutive wrong solutions, the system replaces the *Minimum* heuristic with the *Maximum*, indicating exercises that require skills that this student is already high. This approach is adopted to encourage the student to continue doing the exercises. When all student skills are considered high, the system randomly recommends problems.

V. Case Study

A case study was carried out, in a period of 3 months, in a class of 18 beginner students of the Information Systems course enrolled in the discipline Algorithms and Data Structures I. Due to the number of participants, and the low number of submissions, the heuristic *Minimum* was adopted for the case study. That is, to select the problems in the recommendation zone, the student's lowest abilities were observed and the problems that could better develop these abilities were indicated. Thus, in this case study, lower skills have greater relevance.

Initially, students registered on the *beecrowd* platform. Afterwards, they filled out a form on *Google Forms* informing their identification of *beecrowd*, e-mail, registration number and institution. With this information, students were registered in the Primary recommender system. A tutorial for accessing the recommendation system and using the *beecrowd* platform was made available. In general, students access the system, via *web*, through a *login* (e-mail) and password (enrollment).

Next, the problem is indicated to the student in the recommended system environment and, by clicking on the problem ID, the student is directed to the platform website, where the student accesses the problem statement and the submission area. The student develops the problem solution and submits it on the site *beecrowd*, which evaluates the solution and gives automatic feedback to the student, as shown in Fig. 2.

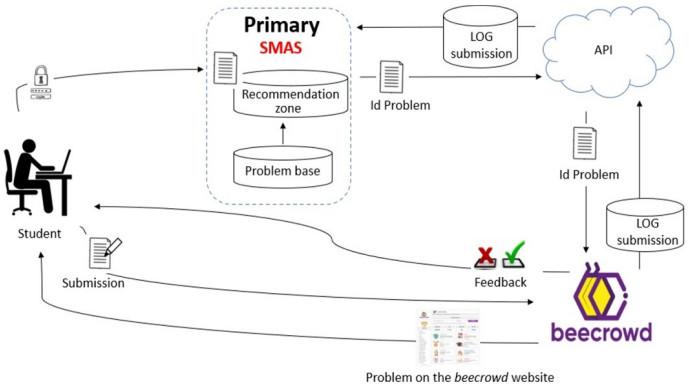

Fig. 2. Case study methodology

A. Results and Discussion

There were 300 interactions, including 136 submissions with correct solutions, 31 incorrect solutions, and 133 skips, when students chose not to solve the recommended problem. Table I shows the relationship of the students with the amounts of correct answers, errors, skips and total submissions. Besides, Table II presents a relationship between students and the respective final values of their abilities.

As can be seen, student 1 was the one who interacted the most, making 162 interactions, where there were 72 problem skips, 80 correct solutions, and 10 wrong solutions. Despite the high number of interactions and correct solutions, this student had a significant drop in his abilities. This occurred because there was a high number of questions that the student did not solve (skipped). As an unsolved problem is considered an error, the student ends up suffering a greater penalty. Figure 3 shows the skill evolution graph of student 1.

The high penalty occurred because it is understood that if the student chose not to solve it, it means that this student

TABLE I
RELATIONSHIP OF STUDENTS WITH THE RESPECTIVE AMOUNTS OF SUCCESSES, MISSES AND SKIPS

User ID	Successes	Errors	Skips	Total
1	80	10	72	162
2	8	6	11	25
3	10	0	9	19
4	13	2	1	16
5	5	1	8	14
6	4	0	8	12
7	2	0	6	8
8	1	2	5	8
9	2	1	4	7
10	5	1	7	7
11	3	1	1	5
12	1	2	1	4
13	1	1	1	3
14	0	1	1	2
15	0	1	1	2
16	1	0	1	2
17	0	1	1	2
18	0	1	1	2
Total	136	31	133	300

TABLE II
RELATIONSHIP OF STUDENTS WITH THEIR RESPECTIVE SKILL VALUES

User ID	Data Analyze	Data Representation	Abstraction
1	858.601	961.592	926.167
2	938.238	957.719	968.622
3	1055.301	1115.796	1099.600
4	1160.090	1178.341	1198.086
5	1001.206	1028.181	1007.245
6	1032.560	1021.307	1022.739
7	1060.481	1045.141	1046.541
8	1035.968	1026.212	1018.675
9	1071.265	1061.423	1057.443
10	1111.425	1108.210	1128.471
11	1103.523	1107.714	1103.475
12	1065.360	1081.712	1070.670
13	1086.224	1091.345	1081.678
14	1072.688	1079.978	1073.184
15	1073.202	1079.542	1073.126
16	1092.918	1095.188	1096.163
17	1059.480	1090.372	1072.740
18	1066.814	1084.262	1079.096

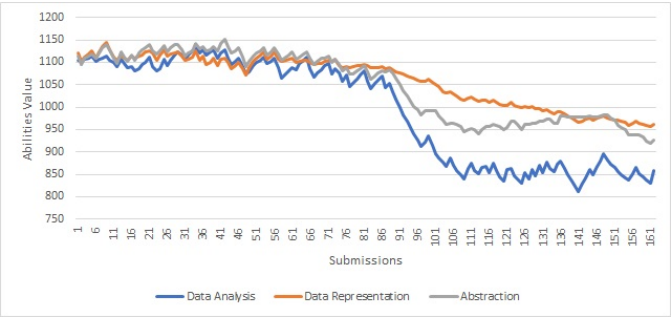

Fig. 3. Evolution of Student 1 abilities

either does not know how to start the solution to this problem. This penalty is analogous to an assessment activity in which the student gets 49% of the questions right (80 right solutions from 162 submitted solutions), and the others are incorrect or unanswered. The issue of skill loss in case of error is a characteristic of the Elo model that, when considering a duel between subject and object, the winner increases his skills, and the loser decreases them.

The student that had the highest ability values was Student 4. From 16 problems that were proposed, 13 submissions were correct, 2 submissions were incorrect and there was only 1 skip. Figure 4 presents the graph of the evolution of skills of this student. Even with this number of correct answers, their skills did not have an expressive increase, which is justified by the skills being updated on the scale based on the success expectation (higher increase when expectations are broken and, on a contrary, smaller increase), the relevance of the skill and a constant. In order to increase the skill increment and decrement scales, an adjustment can be made, raising the constant k in the Elo model, see (6). In this case study, the constant k used is equal to the literature.

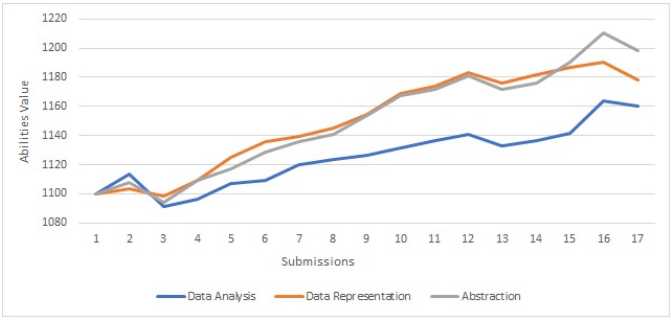

Fig. 4. Evolution of Student 4 skills

Overall, students lowered skill scores because many of the recommended problems were not solved or skipped, being treated as an error, aggravating the skill drop more significantly. Figure 5 shows the graph representing the percentages of correct, incorrect and unsolved problems.

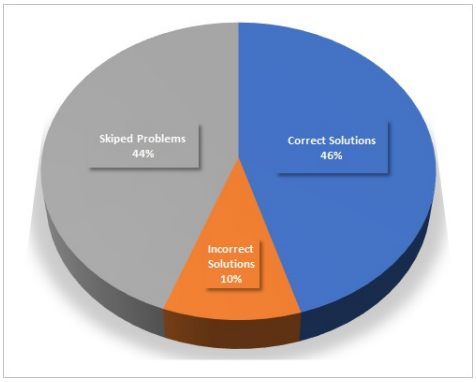

Fig. 5. Result of recommended problems

Although most of the problems recommended by the Primary were solved correctly, the percentage of unsolved problems that students chose not to solve is still high. With each unanswered interaction, there is a decrease in the student's

abilities, and an increase in the difficulty of the unsolved problem. It is believed that this process is generating a false value in the problem difficulties. Therefore, it will be better investigated in the next step of this research.

Furthermore, it is also believed that the high number of unsolved problems is related to the lack of understanding of what is being asked in the problem, since many of the problem statements from *beecrowd* platform are complex; thus, not always easy to understand. Therefore, soon, it is intended to conduct a second experiment where the problems will all be generated and classified by an expert, avoiding problems with ambiguous questions.

VI. Conclusion

In the present work, a recommender system integrated into an Online Judge platform for programming problems was presented. The SMAS model was incorporated into the recommender system, which considers the multiple abilities of students and problems. The recommender system was associated with the *beecrowd* Online Judge platform, which has a repository of programming problems and an automatic evaluation system. Four recommendation heuristics have been implemented (*Minimum*, *Maximum*, *Average* and *Random*) and were used according to the student's abilities or the use objectives of the recommender system. Therefore, a case study was carried out by a group of students enrolled in the discipline of Algorithms and Data Structures I of the Information Systems course. The analyzed results indicate that most of the student's interactions with the recommended problems received a correct answer; that is, in most of the recommendations, the system indicated adequate problems for the students according to their abilities. However, there was a high number of unanswered interactions, when students chose to skip the next problem. One of the hypotheses for many unanswered interactions is the complexity of the problem statement of the *beecrowd* platform.

In the case of students who got the most submissions correct, the scale of the increase was not surprising. This occurred due to the way in which the model updated the skills, considering the success expectation, which increases/decreases considerably when there is an expectation break, and little when the answer is the expected one; besides, being guided by a constant and relevance of the skills involved. Therefore, student's skill scores were greatly affected when they skipped the problems. Such a decision leads to the loss of student's skills and an increase in problem difficulty, because the model penalizes the loser and rewards the winner, and; therefore, in the case of a skip, the problem is considered a winner. As a solution to adjust it, the *skip* can be treated as a second class of error, with reduced penalization. In future work, some adjustments will be made to the model that guides the recommender. One of them is a study on the value that the constant should receive in cases of unresponsive interactions, so that the student is not penalized so much in this case. Moreover, another case study will be carried out with a class with a larger number of students and for a longer period of time, allowing to implement the heuristics in the recommender system. It is believed that the recommender system based on the SMAS model allows a better recommendation by analyzing each student's skill and the skills involved in the learning objects, customizing the exercises for each learner, and stimulating their abilities.

References

[1] G. Falckembach and F. Araujo, "Aprendizagem de algoritmos: dificuldades na resolução de problemas," *Proceedings Sulcomp*, vol. 2, 2013.

[2] V. Barr and C. Stephenson, "Bringing Computational Thinking to K-12: What is Involved and What is the Role of the Computer Science Education Community?" *Acm Inroads*, vol. 2, no. 1, pp. 48–54, 2011.

[3] F. B. Baker, *The basics of item response theory*. ERIC, 2001.

[4] D. F. de Andrade, H. R. Tavares, and R. da Cunha Valle, "Teoria da Resposta ao Item: conceitos e aplicações," *ABE, Sã Paulo*, 2000.

[5] R. Pelánek, "Applications of the elo rating system in adaptive educational systems," *Computers & Education*, vol. 98, pp. 169–179, 2016.

[6] A. Prisco, R. Santos, S. B. N. Tonin, and J. Bez, "Using information technology for personalizing the computer science teaching," in *2017 IEEE Frontiers in Education Conference (FIE)*. IEEE, 2017, pp. 1–7.

[7] A. P. Vargas, R. A. P. dos Santos, J. Bez, N. Tonin, and S. S. da Costa Botelho, "Um modelo de mediação pedagógica para ambientes massivos," *RENOTE-Revista Novas Tecnologias na Educação*, vol. 17, no. 1, pp. 93–102, 2019.

[8] E. P. Pimentel, V. F. de França, R. V. Noronha, and N. Omar, "Avaliação contínua da aprendizagem, das competências e habilidades em programação de computadores," in *Anais do Workshop de Informática na Escola*, Campinas, 2003, pp. 533–544.

[9] A. Gomes, C. Areias, J. Henriques, and A. J. Mendes, "Aprendizagem de programação de computadores: dificuldades e ferramentas de suporte," *Revista Portuguesa de Pedagogia*, pp. 161–179, 2008.

[10] R. D. Pea and D. M. Kurland, "On the cognitive effects of learning computer programming," *New Ideas in Psychology*, vol. 2, no. 2, pp. 137–168, 1984.

[11] J. Wing, "Computational thinking," *Communications of the ACM*, vol. 49, no. 3, pp. 33–35, 2006.

[12] R. L. de Souza, F. Z. Ferreira, and S. S. da Costa Botelho, "Proposta para avaliação de códigos fonte com tf-idf," in *Anais do XXXI Simpósio Brasileiro de Informática na Educação*. SBC, 2020, pp. 112–121.

[13] V. Barr and C. Stephenson, "Bringing computational thinking to k-12: what is involved and what is the role of the computer science education community?" *Acm Inroads*, vol. 2, no. 1, pp. 48–54, 2011.

[14] S. C. Cazella, E. B. Reategui, M. Machado, and J. L. V. Barbosa, "Recomendação de objetos de aprendizagem empregando filtragem colaborativa e competências," in *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)*, Florianópolis, Nov. 2009.

[15] S. C. Cazella, P. Behar, D. Schneider, K. K. da Silva, and R. Freitas, "Desenvolvendo um sistema de recomendação de objetos de aprendizagem baseado em competências para a educação: relato de experiências," in *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)*, Rio de Janeiro, 2012.

[16] S. Wasik, M. Antczak, J. Badura, A. Laskowski, and T. Sternal, "A survey on online judge systems and their applications," *ACM Computing Surveys (CSUR)*, vol. 51, no. 1, pp. 1–34, 2018.

[17] R. E. Francisco, A. P. L. Ambrósio, C. X. P. Junior, and M. A. Fernandes, "Juiz online no ensino de cs1-lições aprendidas e proposta de uma ferramenta," *Revista Brasileira de Informática na Educação*, vol. 26, no. 03, p. 163, 2018.

[18] C. V. Giordano, L. N. de Lira, C. Langhi, and M. D. Feitosa, "Tecnologia de apoio ao ensino e aprendizagem de programação em graduações tecnológicas profissionais: Juiz on-line," *Boletim Técnico do Senac*, vol. 47, no. 2, 2021.

[19] A. P. Vargas, R. Santos, M. Neves, D. Teixeira, and S. S. da Costa Botelho, "Sistema de recomendação baseado no elo para problemas de pré-cálculo: Um experimento com calouros universitários," in *Anais do XXXI Simpósio Brasileiro de Informática na Educação*. SBC, 2020, pp. 1213–1222.

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

[21] D. F. de Andrade, H. R. Tavares, and R. da Cunha Valle, *Teoria da Resposta ao Item: conceitos e aplicações*. São Paulo: ABE - Associação Brasileira de Estatística, 2000.
[22] L. Pasquali, *TRI–Teoria de Resposta ao Item: Teoria, procedimentos e aplicações*. Editora Appris, 2018.
[23] R. T. Nojosa, "Teoria da Resposta ao Item (TRI): modelos multidimensionais," *Estudos em Avaliação Educacional*, vol. 1, no. 25, pp. 123–166, 2002.
[24] M. D. Reckase, "Multidimensional Item Response Theory," *Handbook of statistics*, vol. 26, pp. 607–642, 2006.
[25] J. Y. Park, F. Cornillie, H. L. van der Maas, and W. Van Den Noortgate, "A multidimensional IRT approach for dynamically monitoring ability growth in computerized practice environments," *Frontiers in Psychology*, vol. 10, 2019.
[26] J. H. Rossler, "O desenvolvimento do psiquismo na vida cotidiana: aproximações entre a psicologia de alexis n. leontiev e a teoria da vida cotidiana de agnes heller," *Cadernos Cedes*, vol. 24, pp. 100–116, 2004.
[27] A. Prisco, R. Santos, S. Botelho, N. Tonin, and J. Bez, "A multidimensional Elo model for matching learning objects," in *2018 IEEE Frontiers in Education Conference (FIE)*. IEEE, 2018, pp. 1–9.