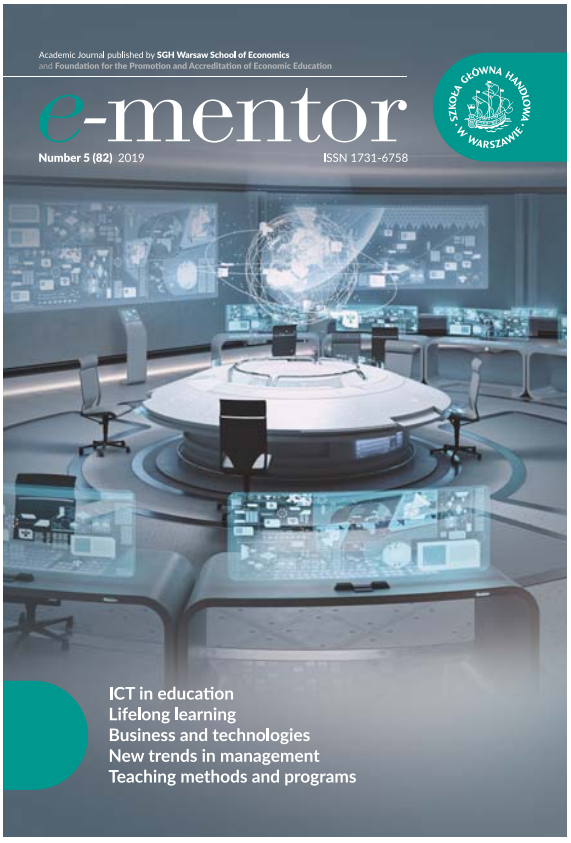
e-mentor

DWUMIESIĘCZNIK SZKOŁY GŁÓWNEJ HANDLOWEJ W WARSZAWIE

WSPÓŁWYDAWCA: FUNDACJA PROMOCJI I AKREDYTACJ KIERUNKÓW EKONOMICZNYCH



2019, nr 5 (82)

Pankiewicz, M. i Bator, M. (2019). Elo Rating Algorithm for the Purpose of Measuring Task Difficulty in Online Learning Environments. *e-mentor, 5*(82), 43–51. https://doi.org/10.15219/em82.1444

Elo Rating Algorithm for the Purpose of Measuring Task Difficulty in Online Learning Environments

Maciej Pankiewicz[*]

Marcin Bator[*]

Abstract

The Elo rating algorithm, developed for the purpose of measuring player strength in chess tournaments, has also found application in the context of educational research and has been used for the purpose of measuring both learner ability and task difficulty. The quality of the estimations performed by the Elo rating algorithm has already been subject to research, and has been shown to deliver accurate estimations in both low and high-stake testing situations. However, little is known about the performance of the Elo algorithm in the context of learning environments where multiple attempts are allowed, feedback is provided, and the learning process spans several weeks or even months. This study develops the topic of Elo algorithm use in an educational context and examines its performance on real data from an online learning environment where multiple attempts were allowed, and feedback was provided after each attempt. Its performance in terms of stability of the estimation results in two analyzed periods for two groups of learners with different initial levels of knowledge are compared with alternative difficulty estimation methods: proportion correct and learner feedback. According to the results, the Elo rating algorithm outperforms both proportion correct and learning feedback. It delivers stable difficulty estimations, with correlations in the range 0.87–0.92 for the group of beginners and 0.72–0.84 for the group of experienced learners.

Keywords: Elo, rating algorithm, task difficulty, online learning environment, learner feedback, proportion correct

There are several methods of estimating task difficulty and several possible applications of them. The process of knowledge testing may be shortened by dynamically selecting tasks according to the learner's ability. There is a whole research area of Computerized Adaptive Testing (CAT) with multiple models developed to shorten the process of knowledge assessment. There are also other possible applications, e.g., within online learning environments, to adjust the difficulty of consecutive items presented to the learner. The aim is to deliver items that are not too difficult, as it may cause learner's frustration. On the other hand, if the task is too easy, the motivation of learners may decrease.

There are several Item Response Theory (IRT) models used within the CAT area. However, the need for calibration of item banks, computational requirements, and complex implementation make their application within online learning environments problematic. Therefore, alternative methods of task difficulty estimation are evaluated, e.g., rating algorithms originally developed and implemented in other fields. The usage of the Elo rating algorithm has already found examples within the educational context, and this research is the authors' contribution to that field.

Measuring task difficulty

The problem of estimating task difficulty within an educational context refers not only to the summative assessment (Wauters, Desmet, & Van den Noortgate, 2012), but is increasingly present in the context of online learning environments with formative assessment approaches (Chen, Lee, & Chen, 2005; Klinkenberg, Straatemeier, & van der Maas, 2011; Kortemeyer, 2014; Pelánek, Papoušek, Řihák, Stanislav, & Nižnan, 2017). In the context of knowledge testing (summative assessment), several models have been developed for the purpose of Computerized Adaptive Testing (CAT), assessing the problem of task difficulty and learner ability. The Item Response Models (IRT) of CAT have been used for adaptive item sequencing with the main

[*] Warsaw University of Life Sciences, Poland

aim of shortening the process of knowledge testing (de Ayala, 2008; Veldkamp & Sluijter, 2019). On the other hand, online learning environments utilize the formative assessment approach and are focused rather on providing feedback than on knowledge testing. The increasing popularity of online learning platforms encourages the introduction of new learning concepts, and these environments may benefit from adaptive item sequencing (Wauters, Desmet, & Van den Noortgate, 2010). The platforms may utilize models for estimating task difficulty, adaptively delivering personalized learning content dependent on the learner's current ability.

Several aspects increase the difficulty of implementing IRT models developed for the purpose of adaptive knowledge testing in the context of online learning platforms. The main assumption of these models is that the skill level of the learner is a constant, as measured at a certain moment of time. Another issue is related to the computational requirements of IRT models. With the growing amount of task response data, satisfying the requirements of recalculation task difficulty and learner ability becomes impossible in real time. Simulation studies (Verschoor, Berger, Moser, & Kleintjes, 2019) have been performed on a database consisting of 300 tasks. According to the study, the response time increases to 2 seconds for 20 observations per task and to 140 seconds for 200 observations per task. However, both these issues are in contradiction to the requirements of online learning environments, where it is expected that knowledge level develops over time, and the updating of estimations for both task difficulty and learner ability is provided on-the-fly.

Several methods of difficulty estimation (Klinkenberg et al., 2011; Wauters et al., 2012; Pelánek et al., 2017; Morrison, 2019) have already been examined as an alternative to IRT models to meet the requirements of online learning platforms and to satisfy the on-the-fly calibration requirements of these platforms. It has been found that rating algorithms, e.g. Elo (Elo, 1978) or Glicko (Glickman, 2001) may deliver difficulty estimations of acceptable accuracy (Pelánek et al., 2017), with lower computational requirements and simpler implementation.

Rating algorithms have been implemented in several online learning platforms for the purpose of estimating task difficulty, with examples of simple multiple-choice or open-ended tasks in the areas of mathematics (Klinkenberg et al., 2011), geography (Pelánek et al., 2017) and foreign languages (Wauters et al., 2012). The simplicity of the task types results from the fact that solving a mathematical task requires the entry of the correctly calculated number (Klinkenberg et al., 2011). Testing the knowledge of geography facts requires the entry of the correct name of a country or capital (Pelánek et al., 2017).

This study introduces the problem of solving very complex assignments: programming tasks. The complexity of the programming task results from the necessity to write the lines of programming code defining the algorithm that solves a given problem. The learner uploads the created code to the platform and receives feedback on the prepared solution. Multiple attempts are allowed, which should encourage the uploading of improved versions of the code if the previous submission was incorrect.

Online learning environments for a broader audience may benefit from the implementation of methods for evaluating task difficulty and utilizing the estimations in order to adaptively deliver consecutive items. However, the introduction of these methods in smaller e-learning courses across the educational system may also be beneficial. It may provide an insightful overview of the course content difficulty levels and help in better understanding the deficiencies in student knowledge. The involvement of complex methods such as IRT models is a challenge due to the implementation requirements and high computational demands. Therefore, methods delivering accurate estimations where the implementation requirements are lower are attracting the attention of researchers and are a subject for this discussion.

This article presents the approach of utilizing the Elo rating algorithm in the context of an online learning environment, where multiple attempts are allowed, feedback is provided after each attempt and the assignment is demanding: programming tasks, requiring the learner to write code containing algorithms to solve problems of varying difficulties. It can be seen that the Elo rating algorithm provides consistent estimations of task difficulty in the compared learning periods for both groups of learners: beginners and experienced. Additionally, it can be seen that the certainty of estimations increases for all the analyzed methods if the reliability of the assignment to the group according to the level of knowledge increases.

Elo rating algorithm

The aim of the Elo rating algorithm (Elo, 1978) is to estimate a player's strength in two-player tournament games. It was developed to be implemented within chess tournaments and has since been adopted by many other sports organizations, e.g. hockey, table tennis and basketball. It has also found implementations in the context of online learning environments and has proved to be useful in estimating both task difficulty and learner ability (Klinkenberg et al., 2011; Wauters et al., 2012; Pelánek et al., 2017). According to the research, the estimations delivered by the Elo rating algorithm are accurate enough for operational use within online learning environments. Additionally, implementation requirements and computational demands are low, allowing it to be used as an on-the-fly solution in environments with large numbers of system users and available assignments (Verschoor et al., 2019).

The rating of learners in an online environment is calculated based on the assignment submission outcomes. It is subject to change after every submission.

The formula for calculating a new ranking is as follows:

$$R_n = R + K(O - P) \quad (1)$$

Where: R_n is the new value of the rating, R – the actual rating, O – submission outcome (1 – fully correct response, 0 – incorrect response), P – probability of submitting the fully correct response and constant K – the optimal value, which is subject to calculations, and for chess tournaments the value is often 32.

The probability of submitting fully correct response P for a learner is calculated with the following formula:

$$P = \frac{1}{1 + 10^{\frac{R_O - R_p}{400}}} \quad (2)$$

Where R_p is the rating for a learner and R_o is the rating for the assignment.

There have been several extensions to the Elo rating algorithm proposed in the literature. Several studies have utilized the basic version of the algorithm and have evaluated the K value experimentally (Wauters et al., 2012; Antal, 2013). However, there are other approaches, e.g. calculating the K value depending on the number of assignments solved by a learner (Papoušek, Pelánek, & Stanislav, 2014; Wauters, Desmet, & Van den Noortgate, 2011).

In order to perform calculations with the Elo rating algorithm, the input data consists of the list of all attempts recorded by the system containing the following columns: 1) learner, 2) task, 3) round and 4) score. The order of the games in the table reflects the consecutive order of attempts recorded by the system.

The *learner* column contains the learner ID and the *task* column denotes the ID of the task in the system. Possible values of the *score* column are: 1 – the learner solves the task completely (win) and 0 – the learner fails to solve the task (loose). The involvement of the round parameter is reasonable in situations where multiple tournaments may be divided by periods of inactivity. In the context of the learning environment, all submissions are considered to occur within one tournament (Wauters et al., 2012; Pelánek et al., 2017) and this study also utilizes this approach.

Proportion correct

Proportion correct (PC) is a simple measure, calculated as the number of correct attempts on task divided by the total number of attempts on the assignment.

The difficulty of the *i-th* task is therefore calculated as:

$$\hat{b}_i = 1 - \frac{n_i}{N_i} \quad (3)$$

Where n_i is the number of correct attempts and N_i the number of total attempts on the *i-th* task. The greater the number of correct attempts achieved on the task in total, the lower the difficulty of that task.

According to Wauters et al. (2012), PC may generate accurate estimations if administered to 200–250 learners. The accuracy of the method is very high according to several studies (Wauters et al., 2012; Antal, 2013; Morrison, 2019).

Learner feedback

Learner feedback (LF) is another simple measure, based on the concept of crowdsourcing or collaborative voting. Difficulty estimations are based on learner estimation of difficulty. Learners judge the assignment on the rating scale and the difficulty estimation $\acute{x}$ is the mean of n collected responses x_i calculated for each task, as:

$$\bar{x} = \frac{1}{n} \sum_{i=0}^{n} x_i \quad (4)$$

For the purpose of the present analysis, the learners rated the difficulty of assignments on the 5-point Likert scale in the range: 1 – very easy task, 5 – very difficult task. The learners were not obliged to rate the tasks, their responses were optional. Additionally, no attempt on the task was required prior to rating the task, and the learners could rate the task anytime, even without a trial.

Obtaining accurate estimations based on LF depends on the number of collected responses from the learners (Chen et al., 2005; Wauters et al., 2012). The accuracy of the difficulty estimations performed by LF increases with the number of learners willing to share their opinion on task difficulty. However, collecting a sufficient number of responses is an organizational problem and for most online courses with smaller numbers of learners this may take some time. If the responses are collected from a sufficient number of learners, the method provides high accuracy in the estimations (Wauters et al., 2012). The number of collected learner responses collected in terms of this study, broken down into modules, is presented in Table 1.

Methodology

This section presents specific issues related to the construction of the study, the process of data gathering and the implementation of the utilized algorithm, i.e. the online learning environment, the complexity of the programming task utilized for the study and details related to the groups compared within the analysis.

Data

The data was collected during a programming course taught at the Faculty of Applied Informatics and Econometrics at the Warsaw University of Life Sciences. It encompassed two periods: P1 was the winter semester 2017/2018 and P2 the winter semester 2018/2019. First semester students at the faculty

take the mandatory course of Programming Fundamentals and learn the basic concepts of programming languages in a traditional way, with the support of a university Moodle e-learning platform, where the main course material is available.

Additionally, the students were provided with access to the *RunCode* online learning environment, an internet application available at runcodeapp.com. Use of the additional online learning environment was optional and was not graded; however, the majority of the students decided to use the platform on a regular basis.

The data set contains information about 237 learners that uploaded 33 619 assignment solutions for 76 tasks divided into 7 modules available on the platform. The difficulty of tasks varied, with both very easy and very demanding tasks. The available modules with their respective number of tasks are presented in Table 1.

Table 1. Modules with the respective numbers of tasks and task ratings

Module	Tasks	Ratings
1. Types	11	657
2. Conditional statement	10	493
3. Recursion	10	425
4. Loops	11	356
5. Recursion on Arrays	8	160
6. Loops on Arrays of numbers	12	262
7. Loops on Arrays of characters	14	187
Sum	76	2540

Source: authors' own work.

At the beginning of the course the students were asked to self-evaluate their level of knowledge concerning programming fundamentals on a 5-point Likert scale in the range: 1 – lack of knowledge, 5 – very good knowledge. For the following comparison, those students declaring no or very little previous experience with programming (responses 1 or 2) were assigned to the group of beginners, A (n=152), while the remaining students declaring previous experience with programming were assigned to the experienced group, B (n=85). The number of users and task submissions (attempts), divided into modules for each group, is presented in Table 2. The initial number of students enrolled in the course was equal during both periods; however, more students decided to use the application in period P2. This may result from the very positive feedback the application received from students using it in the first period. The new students were encouraged to use the application by their older colleagues. Therefore, the students in period P2 started using the application earlier, on average, which may to some extent explain the differences described in the following sections.

Several aspects of the summary presented in Table 2 are important according to this study. There were more learners from the group of beginners using the application, which may be for the following reason. About the half of the students of the first semester at the faculty declared having very low or no previous experience with programming. These students were in general more eager to use the application as it allowed them to experience more uncertainty related to the course material than the students from the experienced group. The students from the experienced group might have felt more confident about their level of knowledge and therefore were less interested in using additional resources.

It may also be observed that the engagement of the students across all modules was stable. The number of attempts across all modules varied slightly; however, the average number of submissions was high for every module. This effect may result from the implementation of gamification elements, improving the engagement and motivation of the system users (Wang & Eccles, 2013; Pankiewicz, 2016).

Programming task

The aim of the programming task available on the platform was to create a small program, a function to return the correct value for any given input. In contrast to popular and simple question types available

Table 2. Comparison of the number of users, total and average number of attempts (Att.) within modules (Mod.) in groups A and B across two analyzed periods: P1 and P2

Mod.	P1-A			P1-B			P2-A			P2-B		
	Att.	Users	Avg	Att.	Users	Avg	Att.	Users	Avg	Att.	Users	Avg
1	903	53	17.0	383	25	15.3	2440	87	28.0	1014	43	23.6
2	1172	48	24.4	468	23	20.3	2455	79	31.1	904	39	23.2
3	1409	53	26.6	600	29	20.7	2596	82	31.7	1245	42	29.6
4	1527	52	29.4	554	24	23.1	2970	80	37.1	1073	37	29.0
5	891	45	19.8	443	22	20.1	1610	64	25.2	637	35	18.2
6	1116	48	23.3	389	23	16.9	2181	70	31.2	745	31	24.0
7	1021	48	21.3	476	25	19.0	1676	61	27.5	721	33	21.8

Source: authors' own work.

on e-learning platforms, such as multiple-choice or open-ended questions, the probability of guessing the correct answer in a reasonable number of attempts was very low for the programming task.

The purpose of the programming task was to create a logical sequence of operations, an algorithm, that solves a certain problem of various levels of difficulty. Consider the simple problem of checking whether two numerical values passed to a function are not equal. The learner was instructed that the aim of the task was to "create the function *bool IsNotEqual (int x, int y)* to return *false* if the numbers x and y are equal, otherwise return *true*."

In order to submit the solution and receive a *score* (0–100%) and *feedback* from the system, the learner created a code containing the required signature of the function (name, parameters and type of the returned value) and code that returned the logical (Boolean) value of *true* or *false*, depending on the numbers passed to the function. After submitting the code to the platform, the code was actually executed by the test runner in order to verify if the function returned the expected results.

The *score* was calculated as the proportion of tests that ended in success to the overall number of tests performed on the code. The tests were defined by a lecturer. The number of tests depended on the complexity of the assignment, but usually varied between 5 and 10, which was enough to test simple functions. The purpose of the multiple tests was to assure that an uploaded solution returned the correct result for every potential combination of parameters. There could be later tests defined for the *IsNotEqual* function, to validate the correctness of its implementation: checking if the function returned *true* when the parameters x and y were not equal, e.g. $x = 2$ and $y = -2$, and if it returned *false* when the parameters x and y were equal, e.g. $x = 2$ and $y = 2$. In order to strengthen the reliability of the evaluation, further tests with several different parameter values could be defined.

Multiple uploads of the solution were allowed. The system provided feedback after every submission, involving three elements: 1) formal errors (if the programming fundamental errors were mostly syntax mistakes) returned by the compiler, e.g. the line of the code did not end with a semicolon and therefore the code could not be compiled; 2) warnings returned by the compiler, e.g. defining a variable that was not used in the program; and 3) test results returned by the test runner containing information about the effects of executing the submitted code.

If the compiler detected formal errors in the uploaded code, e.g. a missing semicolon, execution of the submitted code was impossible and therefore the submission received 0 points. If there were no errors in the submission, unit tests were then executed on the function by the test runner in order to check that the function returns the expected value for each defined input parameter set.

If the information returned by the compiler contained only warnings, the code was still executed by the test runner; however, the submission did not receive a 100% score. Each warning detected by the compiler lowered the grade by 1%.

The third component of the feedback information presented all information about the *parameters* used to execute the unit test, the *expected result* and *actual result* returned by the function submitted by the learner.

Compared groups

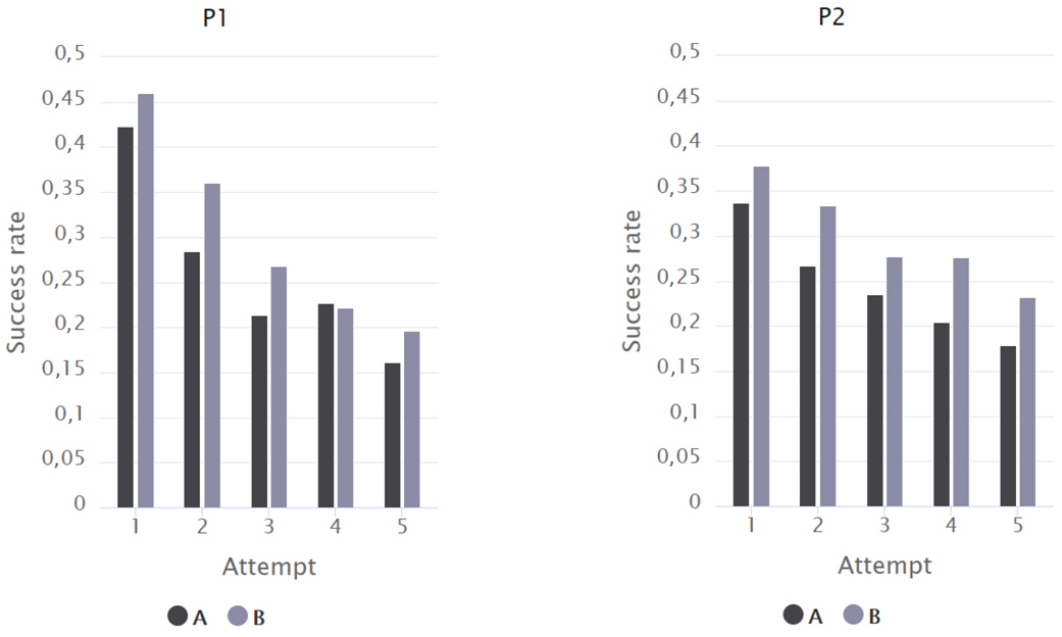
Each group, the beginners and experienced learners, had different initial levels of knowledge. This difference clearly led to other differences, such as in the average number of attempts needed to successfully solve each task. However, this study focuses on groups of learners of the same declared initial level of knowledge. The task difficulty estimations obtained from the group of beginners in period P1 was compared to the results obtained from the group of beginners in period P2. The same comparisons were performed on the groups of experienced learners. As a result, it can be shown that groups of the same declared initial level of knowledge presented significant differences in the number of attempts needed to completely solve the assignment across the analyzed periods. Secondly, it can be demonstrated how this difference impacted the quality of the difficulty estimations, as performed by the different methods of evaluating task difficulty.

Learners utilize the feedback information in order to upload a corrected version of the solution. The data shows that the tasks were on average very challenging, both for the beginners and the experienced learners. The success rate for all consecutive attempts decreased. Less than half of the first-attempt submissions received a maximum score (100%) for both groups. The success rate in consecutive attempts differed in both compared groups in each of the analyzed periods. In general, the more experienced learners succeeded more often than the beginners, which was as expected. However, the difference was expected to be higher. A moderate difference may occur if learners having previous experience with programming were optimistic in estimating their real level of knowledge. More reliable estimations of knowledge level was expected in the group of beginners. The overview of the success rate for the first five attempts for both groups across analyzed periods is presented in Figure 1.

Not all learners completely solved the tasks where they uploaded at least one solution. There may be different reasons for not attempting to solve a task completely, e.g. skipping to another task that seemed more interesting, lack of time, or decreasing level of motivation after several unsuccessful attempts. The comparison of the average number of attempts needed to completely solve the assignment for both groups in the two analyzed periods is presented in Figure 2.

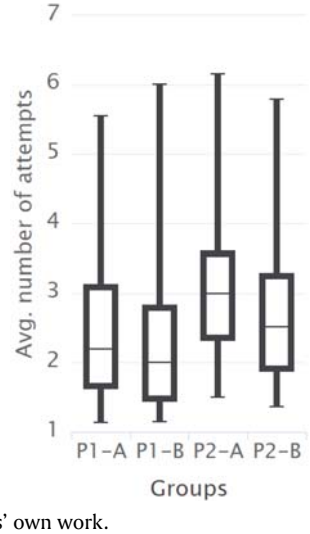
The results met the expectations, as it was observed that the experienced learners (B) needed less attempts to solve the assignments than the beginners (A) (Figure 2). This effect was observed in both compared periods: P1 and P2.

Figure 1. Comparison of the success rate for the first five attempts across periods P1 and P2 for groups A (beginners) and B (experienced learners)



Source: authors' own work.

Figure 2. Comparison of the average number of attempts for the tasks solved correctly across periods P1 and P2 for groups A (beginners) and B (experienced learners)



Source: authors' own work.

There were also statistically significant differences observed between the groups of beginners and experienced learners across both periods in terms of the number of attempts required to solve a task completely. The U Mann-Whitney indicated that the number of attempts in period P1 was significantly lower than in period P2 for both analyzed groups: A and B (p-value < 0.001) (Table 3).

This test is used when the assumptions of the t-test cannot be met – which was the case in this analysis (assumption of normal distribution).

There were significant differences observed between the groups of learners declaring the same level of knowledge for each period. Potential reasons for these differences were explored in the previous sections. The primary difference between the groups is that users in period P2 began to use the application earlier. The tasks available on the platform did not change; however, significant differences explored between analyzed groups suggested that the estimations of task difficulty may also differ significantly. The purpose of the following analysis

Table 3. Comparison of the average number of attempts for tasks solved correctly across periods P1 and P2 for groups A and B with results of the U Mann-Whitney test

	P1	P2	U Mann-Whitney test
Avg. no. of attempts – Group A	n = 2389	n = 3665	
	2.4914	3.1945	$p < 0.001$
Avg. no. of attempts – Group B	n = 1151	n = 1836	
	2.2684	2.774	$p < 0.001$

Source: authors' own work.

is to explore the relationships between differences in the difficulty estimations that result from the fact that the tasks were solved by different learner groups – even if the initial level of knowledge was held constant.

The following chapter compares three methods of difficulty estimation: Elo rating algorithm, proportion correct and learner feedback, in terms of stability of the delivered difficulty estimations.

Results

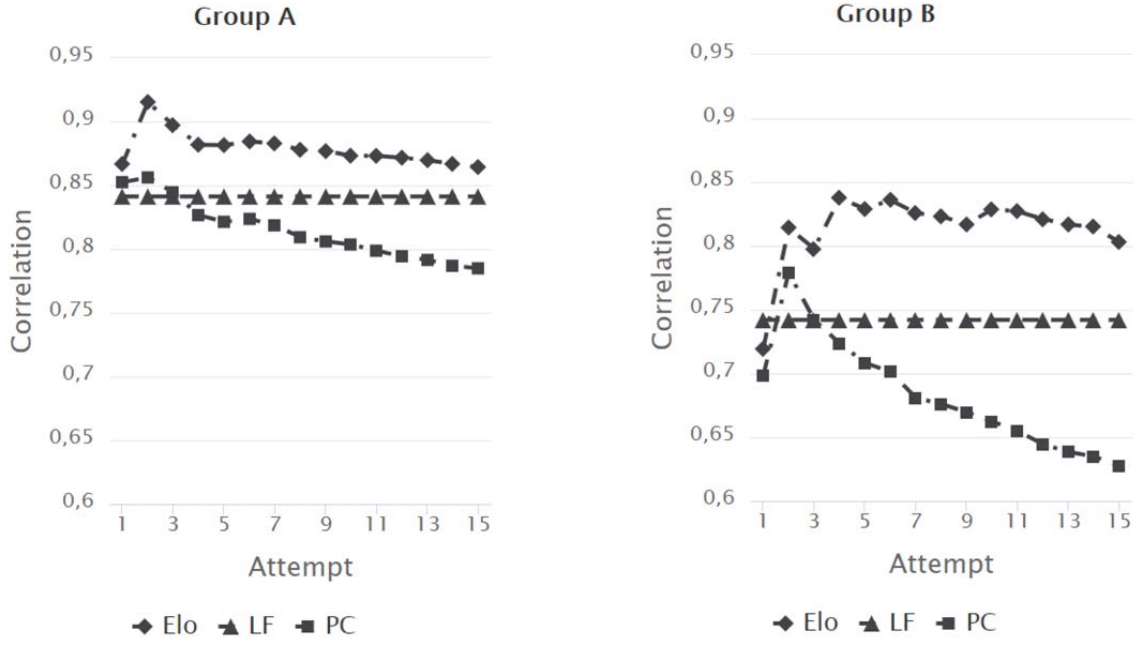
For the purpose of this study, the optimal value of the Elo parameter *K* was evaluated by experiment. The range used for the search was between 1 and 50. The highest correlation between estimations of difficulty in both the analyzed periods for the group of beginners (A) is observed for a *K* value of 11. For the group of experienced learners (Group B), the highest correlation is observed for a K value of 30. The Elo rating algorithm calculations were applied with the PlayerRatings R package (Stephenson & Sonas, 2019) with the default value for the initial rating for both learners and tasks.

The results of the analysis (Figure 3) show that group of beginners (Group A) achieved more stable estimations of task difficulty. The difficulty estimations were less stable for the group of experienced learners (Group B). In both groups, the Elo rating algorithm outperformed the other methods, and was the least sensitive to differences between both analyzed groups across the compared periods.

The Pearson's correlation coefficient was calculated for the estimations of task difficulty in the two learning periods (P1 and P2). The analysis was performed for two groups of learners: beginners (Group A) and experienced learners (Group B). Three methods were compared: Elo rating algorithm, learner feedback (LF) and proportion correct (PC) for a number of attempts ranging from 1 to 15 (Figure 3).

It can be observed that the difficulty estimations from the Elo rating algorithm were more consistent across the analyzed periods (correlation in the range 0.87–0.92 for Group A and 0.72–0.84 for Group B) than LF (0.84 for Group A and 0.74 for Group B) and PC (0.78–0.85 for Group A and 0.63–0.78 for Group B) for both analyzed groups. Elo was less malleable than PC, and this effect has been observed as stable with growing numbers of analyzed attempts. This may result from the main assumption of the rating algorithm and the involvement of the probability estimations in the rating change. While every attempt influenced the PC estimations with the same ratio, its impact on Elo depended on the probability of winning (or loosing) the game. Therefore, the rating change was less dynamic for attempts where the rating of the learner and task was similar and increased with greater distance between learner and task ratings. In terms of this analysis, the increasing number of user's unsuccessful attempts dynamically increased the difference in the PC estimations. For Elo, each additional unsuccessful attempt had less impact on the task rating, as it became less probable that the learner would solve the assignment and the dynamic of the task rating change decreased.

Figure 3. Correlation between task difficulty estimations in periods P1 and P2 for the groups of beginners (Group A) and experienced learners (Group B). Comparison of three methods: Elo rating algorithm (Elo), learner feedback (LF) and proportion correct (PC) for the number of attempts between 1 and 15



Source: authors' own work.

Conclusions

The aim of this research was to compare methods of task difficulty estimations suitable for use within online learning environments for the purpose of adaptive item sequencing. Three methods were compared in terms of stability of the delivered results: Elo rating algorithm (Elo), proportion correct (PC) and learner feedback (LF). The comparison was performed on data from two periods, P1 and P2. Two groups of learners were compared: beginners (A) and experienced learners (B). Assignment to the appropriate group was based on the learner self-assessment.

It has been shown that Elo outperforms the PC and LF methods and delivers the most stable estimations for both compared groups of learners. The task difficulty estimations performed on the group of beginners (Group A) are overall more stable than for the group of experienced learners (Group B). The reasons of this discrepancy may be two-fold: it may be assumed that the self-assessment of the current level of knowledge for students with no or little previous experience is more reliable than for the students declaring more previous experience. Students who already had previous experience with programming may tend to overrate their knowledge level in a self-assessment. The second aspect is related to the difference in the number of students and recorded attempts across both groups. The number of collected attempts for Group B may be too low to deliver stable results.

Although the self-assessment method is a simple and quick method of estimating initial knowledge level, it seems to be more reliable if the declared knowledge level is low or very low, at least in terms of the analyzed course subject with a task that is demanding: a programming assignment. In order to avoid problems with estimating the initial level of learner knowledge, a pretest could be used to verify the self-assessment ratings.

Several of the study assumptions need to be taken into consideration in implementing Elo in other areas of education. Firstly, the compared groups share similarities in terms of previous educational experience. Secondly, the amount of gathered data is sufficient to draw conclusions: although the activity was not mandatory, the level of student engagement was very high. Thirdly, the programming task utilized in the study is demanding. The use of a multiple attempt approach for multiple choice or even mathematical open-ended questions could lead to different outcomes.

Online learning environments may benefit from the implementation of fast methods for estimating task difficulty and learner ability. This study has shown that the implementation of the Elo rating algorithm in the context of a learning platform where the task is demanding, with multiple attempts being allowed and feedback provided, may be a reasonable choice.

References

Antal, M. (2013). On the use of Elo rating for adaptive assessment. *Studia Universitatis Babeş-Bolyai, Informatica*, *58*(1), 29–41.

Ayala, R. J. de (2008). *The Theory and Practice of Item Response Theory*. New York, NY: The Guilford Press.

Chen, C. M., Lee, H. M., & Chen, Y. H. (2005). Personalized e-Learning System Using Item Response Theory. *Computers & Education*, *44*(3), 237–255. https://doi.org/10.1016/j.compedu.2004.01.006

Elo, A. E. (1978). *The rating of chess players past and present*. New York, NY: Arco Publishing.

Glickman, M. E. (2001). Dynamic paired comparison models with stochastic variances. *Journal of Applied Statistics*, *28*(6), 673–689. https://doi.org/10.1080/02664760120059219

Klinkenberg, S., Straatemeier, M., & van der Maas, H. L. (2011). Computer adaptive practice of maths ability using a new item response model for on the fly ability and difficulty estimation. *Computers & Education*, *57*(2), 1813–1824. https://doi.org/10.1016/j.compedu.2011.02.003

Kortemeyer, G. (2014). Extending item response theory to online homework. *Physical Review Physics Education Research*, *10*(1), 010118. https://doi.org/10.1103/PhysRevSTPER.10.010118

Morrison, B. (2019). *Comparing Elo, Glicko, IRT, and Bayesian IRT Statistical Models for Educational and Gaming Data* (Doctoral dissertation, University of Arkansas, Fayetteville). Retrieved from https://scholarworks.uark.edu/etd/3201

Pankiewicz, M. (2016). Data analysis for measuring effects of gamification in e-learning environments. In L. Gómez Chova, A. López Martínez, & I. Candel Torres (Eds.), *Edulearn 16: 8th International Conference on Education and New Learning Technologies* (pp. 7082–7088). Barcelona: IATED. DOI: 10.21125/edulearn.2016.0546

Papoušek, J., Pelánek, R., & Stanislav, V. (2014). Adaptive practice of facts in domains with varied prior knowledge. In J. Stamper, Z. Pardos, M. Mavrikis, & B. M. McLaren (Eds.), *Proceedings of the 7th International Conference on Educational Data Mining* (pp. 6–13). London: EDM 2014. Retrieved from http://educationaldatamining.org/EDM2014/uploads/procs2014/long%20papers/6_EDM-2014-Full.pdf

Pelánek, R., Papoušek, J., Řihák, J., Stanislav, V., & Nižnan, J. (2017). Elo-based learner modeling for the adaptive practice of facts. *User Modeling and User-Adapted Interaction*, *27*(1), 89–118. https://doi.org/10.1007/s11257-016-9185-7

Stephenson, A., & Sonas, J. (2019). R package "Player Ratings". Retrieved from https://CRAN.R-project.org/package=PlayerRatings

Veldkamp, B. P., & Sluijter, C. (Eds.). (2019). *Theoretical and Practical Advances in Computer-based Educational Measurement*. Cham: Springer International Publishing.

Verschoor, A., Berger, S., Moser, U., & Kleintjes, F. (2019). On-the-Fly Calibration in Computerized Adaptive Testing. In B. P. Veldkamp, & C. Sluijter (Eds.), *Theoretical and Practical Advances in Computer-based Educational Measurement* (pp. 307–323). Cham: Springer International Publishing.

Wang, M. T., & Eccles, J. S. (2013). School context, achievement motivation, and academic engagement: A longitudinal study of school engagement using a multidimensional perspective. *Learning and Instruction*, *28*, 12–23. https://doi.org/10.1016/j.learninstruc.2013.04.002

Wauters, K., Desmet, P., & Van den Noortgate, W. (2010). Adaptive item – based learning environments based on the item response theory: possibilities and challenges. *Journal of Computer Assisted Learning*, *26*(6), 549–562. https://doi.org/10.1111/j.1365-2729.2010.00368.x

Wauters, K., Desmet, P., & Van den Noortgate, W. (2011). Monitoring learners' proficiency: weight adaptation in the elo rating system. In M. Pechenizkiy, T. Calders, C. Conati, S. Ventura, C. Romero, & J. Stamper (Eds.), *Proceedings of the 4th International Conference on Educational Data Mining* (pp. 247–252). Eindhoven: EDM 2011. Retrieved from http://educationaldatamining.org/EDM2011/wp-content/uploads/proc/edm2011_paper33_short_Wauters.pdf

Wauters, K., Desmet, P., & Van den Noortgate, W. (2012). Item difficulty estimation: An auspicious collaboration between data and judgment. *Computers & Education*, *58*(4), 1183–1193. https://doi.org/10.1016/j.compedu.2011.11.020

Maciej Pankiewicz (PhD) is an assistant professor at the Institute for Technical Computer Science at the Warsaw University of Life Sciences. He has 10+ years of experience in the implementation and development of e-learning systems in both university and business environments. His main areas of interest concern the use of data analysis methods in the area of educational data mining. ORCID: 0000-0002-6945-0523

Marcin Bator (PhD) is an assistant professor at the Institute for Technical Computer Science at the Warsaw University of Life Sciences. His scientific activities refer mainly to the area of pattern recognition and evolutionary algorithms. ORCID: 0000-0002-6881-3695

We recommend

Olga Robinson, Alistair Coleman, Shayan Sardarizadeh,
A report of anti-disinformation initiatives

Fake news is a global problem that challenges how we share information and perceive the world around us. Evidence of home-grown and foreign online influence operations has caused alarm and concern among politicians and voters. There are fears that democratic institutions and national elections are under threat from mis-, dis-, and mal- information shared on a huge scale online and on social media platforms. Mob lynchings and other violence based on false rumors have turned fake news into an emergency in some parts of the world, costing lives and causing significant problems for societies. This has prompted a number of governments to adopt measures ranging from legislative and legal action to media literacy and public awareness campaigns to fight the spread of disinformation.

In this report, BBC Monitoring's specialist Disinformation Team investigates fake news landscapes around the world and analyses a range of measures adopted by governments to combat disinformation. The study provides a geopolitical context with timely, relevant examples from 19 countries in four continents (with a particular focus on European nations). The team also reports on the European Union because of its size, power, and influence.

Concerns about making just, effective laws to counter fake news are amplified by some countries' creation of legislation which purports to fight disinformation but appears instead to be used to attempt to gain greater control over their media environment and to suppress debate on social media. Our report indicates that there does not currently seem to be any quick fix that would allow governments to curb the spread of disinformation effectively through legislation without prompting criticism.

Excerpts from the report published by the University of Oxford, August 2019.

More information at **https://oxtec.oii.ox.ac.uk/wp-content/uploads/sites/115/2019/08/OxTEC-Anti-Disinformation-Initiatives-1.pdf**