Recommending Mathematical Tasks Based on Reinforcement Learning and Item Response Theory

Matteo Orsoni[1(✉)], Alexander Pögelt[2], Nghia Duong-Trung[3], Mariagrazia Benassi[1], Milos Kravcik[3], and Martin Grüttmüller[2]

[1] University of Bologna, Piazza Aldo Moro 90, 47521 Cesena, Italy
{matteo.orsoni2,mariagrazia.benassi}@unibo.it
[2] Leipzig University of Applied Sciences, Karl-Liebknecht-Straße 132, 04277 Leipzig, Germany
{alexander.poegelt,martin.gruettmueller}@htwk-leipzig.de
[3] Educational Technology Lab, German Research Center for Artificial Intelligence (DFKI), Alt-Moabit 91C, 10559 Berlin, Germany
{nghia_trung.duong,milos.kravcik}@dfki.de

Abstract. Recommending challenging and suitable exercises to students in an online learning environment is important, as it helps to stimulate their engagement and motivation. This requires considering their individual goals to improve learning efficiency on one side and on the other to provide tasks with an appropriate difficulty for the particular person. Apparently, this is not a trivial issue, and various approaches have been investigated in the areas of adaptive assessment and dynamic difficulty adjustment. Here, we present a solution for the domain of mathematics that rests on two pillars: Reinforcement Learning (RL) and Item Response Theory (IRT). Specifically, we investigated the effectiveness of two RL algorithms in recommending mathematical tasks to a sample of 125 first-year Bachelor's students of computer science. Our recommendation was based on the Estimated Total Score (ETS) and item difficulty estimates derived from IRT. The results suggest that this method allowed for personalized and adaptive recommendations of items within the user-selected threshold while avoiding those with an already achieved target score. Experiments were performed on a real data set to demonstrate the potential of this approach in domains where task performance can be rigorously measured.

Keywords: Recommender System · Reinforcement Learning · Item Response Theory · Personalized Recommendation · Math Exercises

1 Introduction

Conventional university teaching methods usually provide uniform learning exercises for the study groups. Depending on the level of knowledge, exercises can differ in the perception of difficulty by students. For optimal support and challenge of students, an individual selection of tasks is needed, which can be made

C. Frasson et al. (Eds.): ITS 2023, LNCS 13891, pp. 16–28, 2023.
https://doi.org/10.1007/978-3-031-32883-1_2

based on various metrics, e.g. the level of knowledge or the desired final grade. Individualized learning tries to stimulate the motivation and engagement of students, taking into account theories like the zone of proximal development [16] and flow [4]. The first provides students with tasks beyond their current ability to scaffold the learning process. The second aims to avoid boredom and frustration if the chosen difficulty level does not correspond with the student's ability. Dynamic Difficulty Adjustment (DDA) mechanism, which originated from computer games, is a technique used to automatically adjust the difficulty of online tasks according to the abilities of the user [3,17], with the goal of keeping the user's attention and engagement. The DDA concept [1] emphasizes the importance of three aspects: the task difficulty (static or dynamic), the user's status (this can include performance or engagement, but also personality and emotions), and the adaptation method, which can be based on rules or data-driven approaches (e.g. probabilistic models, reinforcement learning). Physiologically, user involvement is driven by discovering new knowledge, learning patterns, ideas, and excitement while achieving a particular learning goal [9]. In educational contexts, DDA can ensure that students are presented with tasks suitable for their current level of proficiency, leading to more engaging learning experiences.

One approach to implementing a DDA mechanism is using the Item Response Theory (IRT), a statistical model that estimates an individual's proficiency at a particular task by analyzing their responses to a set of items [5]. This enables to a recommendation of appropriately challenging tasks for the student. However, recommending tasks based on IRT estimates can be suboptimal, as it does not consider the student's learning progress. Therefore, we propose the IRT integration with Reinforcement Learning (RL), which allows for optimizing task recommendations based on the student's past performance.

This study presents a system that utilizes IRT and RL to recommend tasks to first-semester bachelor's degree computer science students taking a mathematics module. Using our proposed method, which employs and compares the Proximal Policy Optimization (PPO) [12], and the synchronous, deterministic variant of the Asynchronous Advantage Actor-Critic [10] algorithm (called A2C), we aim to demonstrate the benefits of personalized task recommendation in the educational settings. In more detail, we incorporated the learner's goals into our recommender system. Literature suggested that specific interventions to set personal academic goals and exam preparation are essential factors contributing to the student's success while in the university [13]. Moreover, goal setting can help students develop a sense of agency, intrinsic motivation, and the ability to manage their learning [11]. We compared the performance of our proposed method to a random baseline, using data from 125 students. The results of our study will provide insight into the effectiveness of using IRT and RL for recommending items in line with the learner's past performance and goals.

In the following, we first reference some related work and background information. Then we present our experiments thoroughly, including the results. Finally, we discuss the outcomes and conclude the paper.

2 Related Work

Computerized adaptive assessment methods in well-structured domains like mathematics have a long tradition of selecting tasks according to the student's ability [15], where structured task description schemes allowed for a detailed analysis of student's errors and on-demand generation of task instances facilitated independent student work. During the recent Corona crisis, professional rule-based adaptive learning systems like bettermarks[1] were very popular.

Recent machine learning approaches address the DDA issue also in other domains if there is a significant question bank and users with different competencies [18], considering even individual difficulty levels. This method can be applied when three conditions are met: a discrete action space exists, a feedback signal is a quantitative measure of difficulty, and a target performance value is selected.

DDA can be achieved using statistical models such as IRT [5]. IRT estimates a learner's proficiency based on their responses to a set of items and has been applied in various educational contexts [7]. However, traditional recommendation approaches may not be suitable in educational settings where a student's learning potential changes over time. Reinforcement Learning (RL) addresses this issue by optimizing task recommendations based on the student's past performance and progress [14]. In recent years, the combination of IRT and RL has been proposed as a solution for recommendation in mathematics and cognitive domains. For example, the authors in [8] suggested using an RL system to recommend items based on the student's ability estimates from an IRT model to improve algebra abilities. Also, the study mentioned earlier [18] used IRT to estimate the student's knowledge and RL to adjust task difficulty.

This work is distinct from the previous approaches in recommender systems that combine RL and IRT. It utilizes IRT to estimate the difficulty of items based on the student's past performance and uses this information to compute the expected total score threshold distribution for mathematical modules. This relevant information allowed to integrate into an RL system of the learner's goal to make recommendations that align with the student's objectives.

3 Background

In Reinforcement Learning (RL), an agent learns to make decisions by interacting with its environment and receiving feedback through rewards or penalties. The agent's goal is to learn a policy mapping from states to actions that maximize the expected cumulative reward over time [14]. In the present work, we used and compared the performances of two popular RL algorithms: the Proximal Policy Optimization (PPO) [12], and the synchronous, deterministic variant (A2C) of the Asynchronous Advantage Actor Critic (A3C) [10]. PPO is designed to improve the stability and efficiency of policy gradient methods. It is an actor-critic algorithm that uses a value function to estimate the expected cumulative

[1] https://bettermarks.com/.

reward for a given policy, and it uses a trust region method to optimize the policy. The basic idea of PPO is to optimize the procedure so that the new policy is close to the previous one but with improved expected cumulative reward [12]. The variant of A3C combines the actor-critic method with the advantage function. The actor-critic process separates the policy, which generates the actions, from the value function, which estimates the expected cumulative reward for a given policy. The advantage function estimates the improvement of taking a given action compared to the average action. The term "synchronous" refers to the method of updating the parameters of the actor and critic networks. All agents update their parameters simultaneously using the same synchronous data. In contrast, in the original asynchronous version, each agent updates its parameters independently using its data [10].

4 Experiments

4.1 Experimental Dataset

This study analyzes a data set collected at Leipzig University of Applied Sciences starting from the winter semester of 2021/22. The data set includes the results of weekly exercises from a mathematics module taken by 125 Bachelor first-year computer science students. To pass the module, students must solve at least 35% of the weekly exercises over the semester. Each weekly practice includes several tasks specific to the topic covered in that week's lecture. The data set also includes solution attempts made after the semester. The tasks differ slightly for each attempt and student but are assumed to have equivalent difficulty and be based on the same concept. To practice the subject matter, students can work on the exercises and subtasks multiple times. Only the most successful attempt will be counted toward the final grade. The assignments are provided through the OPAL learning management system and ONYX testing software, and some tasks allow using the computer algebra system MAXIMA. The data set is separated into tables for student results and task information. To encourage reproducibility and further investigation, we publish the dataset with the implementation codes on our GitHub repository[2].

Result Features

participant An ascending number that anonymously references students
test id References the weekly exercise (test).
test attempt Attempt in which the student solves the weekly exercise
test score Points scored by the student
test pass score Points to pass the weekly exercise
test max score Maximum points of the weekly exercise
test pass Status whether the student has passed the weekly exercise
item id References the actual subtask in a weekly exercise

[2] https://github.com/MatteoOrsoni/ITS2023_Recommending-Math-Tasks.

item attempt Attempt in which the student solves the subtask
item datestamp Timestamp in which the student completed the subtask.
item sessionStatus Represents the status of the subtask. (final - The student has solved the task and submitted his/her answers; pendingSubmission - The student has viewed the assignment but has not responded to it; pendingResponseProcessing - The student has entered answers but has not submitted them; initial - The student has not viewed the assignment)
item duration Time spent on the subtask
item score Points on the subtask scored by the student
item max score Maximum points of the subtask
item candidate responses Answers from the student
item correct responses Correct answers of the subtask
item candidate responses score Scores of the student's answers
item correct responses score (Maximum)-point scores of the subtask
item variables Variable assignments of the subtask execution

Task Features

item id (Equivalent to the result table) references the subtask
is test Status whether the item is a test (tests are groupings of subtaks and are usually equivalent to weekly exercises).
test name Folder name in which the test file is located
item description Tasks description in HTML format

All in all, there are 18576 solutions from a total of 99 different items in a total of 14 modules (including tests for exam preparation) in the data set. Due to the low number of attempts inside some modules and excluding tests for exam preparation, in this study, the analysis focused on 10 modules. On average, the students needed 464 s and achieved an average of 2.18 points per item, with an average maximum score of 3.37 points. Furthermore, students practiced a single item on average 1.85 times, with a maximum of 72 times.

4.2 Framework and Baselines

The IRT models have been implemented by using the mirt: a Multidimensional Item Response Theory Package in R [2], while the RL solutions in Python by using the Stable Baseline 3 [6] library. We compared the two RL solutions (PPO, A2C) with a random baseline procedure. According to this, we ran the environment for 1000 episodes, collecting each reward and averaging at the end. For each episode, the actions were taken randomly into the set of those possible. The averaged reward was then taken as baseline values to be compared to the average reward after 1000 episodes estimated by implementing PPO and A2C algorithms. In the following, we will delve deeper into constructing the item difficulty estimation model and the environment in which the RL algorithms were implemented.

4.3 Difficulty Level

In the present study, an IRT approach is used to estimate the difficulty of items presented to students in a course each week and to create different thresholds based on the sigmoid distribution of the estimated total score (ETS) of the winning IRT model. It allows us to consider the learners' objectives for that particular module. IRT is a statistical procedure that allows for the discovery of a learner's latent trait for a specific concept and the estimation of different parameters (difficulty, discrimination, and guessing) embedded within the item according to the chosen IRT model. Three other IRT models (1PL, 2PL, 3PL) were compared, and the best one was selected using metrics such as the Akaike Information Criterion (AIC), Bayesian Information Criterion (BIC), and log-likelihood. The values of these metrics are summarized in Table 1.

Table 1. AIC (Akaike Information Criteria), BIC (Bayesian Information Criterion), and LL (Log Likelihood). In bold, the models for each module that reached the significant level $p < .05$ among others. * It has not been possible to estimate the parameters due to too few degrees of freedom.

Module	1PL			2PL			3PL		
	AIC	BIC	LL	AIC	BIC	LL	AIC	BIC	LL
1	3452	3550	−1692	**3379**	**3569**	**−1623**	3432	3717	−1617
2	995	1021	−488	**932**	**980**	**−450**	941	1012	−447
3	**533**	**545**	**−262**	533	552	−261	NA*	NA*	NA*
4	**1157**	**1183**	**−569**	1167	1213	−568	1182	1251	−567
5	**515**	**529**	**−253**	521	543	−252	529	562	−252
6	711	728	−350	**704**	**733**	**−342**	713	756	−342
7	844	868	−413	**842**	**884**	**−405**	854	917	−403
8	717	746	−348	**711**	**763**	**−336**	731	809	−336
9	**1059**	**1096**	**−516**	1066	1133	−507	1073	1174	−498
10	**664**	**698**	**−318**	673	735	−310	682	775	−302

The winner over the three possible models was then selected based on the p-value obtained. Only the significantly different model ($p < .05$) from the others was used in further analysis. The estimated total score of the winner IRT model has been then used to estimate the θ value, the correspondent threshold difficulty for that specific module. The thresholds have been chosen arbitrarily except for the first, which was the one that allowed the student to pass the module. Two to four thresholds have been set into the RL solution for each module, according to the number of items (number of possible actions) and the steepness of the sigmoid distribution underlying the estimated total score. Moreover, the IRT solution gave us the values of items' difficulty for each module. These values have been used in the RL environment configuration.

4.4 Reinforcement Learning Environment

The recommender system has been developed as a Markov Decision Process (MDP), consisting of a tuple (S, A, R, P) of elements. The tuple defines the MDP completely, where the agent interacts with the environment. The goal is to find a policy (mapping from states to actions) that maximizes the expected cumulative reward over time. A specific recommender system has a similar MDP representation for each module created. It has been summarized as follows:

1. State Space S: It represents all possible states of the system. It is related to the answering process of the student according to the item presented in the module. Each state or item in the module has been described as a tuple of five elements (d, s, m, dt, t), where:
   (a) d: The difficulty of the module items according to the IRT difficulty estimation.
   (b) s: The score obtained by the student for each item.
   (c) m: The maximum possible score for that specific item.
   (d) dt: The difficulty threshold. This parameter does not change until the end of each episode.
   (e) t: It is the threshold. This parameter does not change until the end of each episode and is strictly related to the difficulty threshold. It is a numeric value corresponding to the score the student has to obtain by summing the score items.
2. Action Space A: It represents all possible actions that can be taken in each state.
3. Reward Function R: It is a function that assigns a numerical reward to each state-action pair (s, a) and is used to evaluate the quality of different policy choices. We included three different numerical rewards in the environment. A reward is related to the $Difficulty$, $Actions$, and $Episode$.
   (a) $Difficulty$: For every action chosen by the agent, that is, for every next item chosen, we wanted to create a function that gave a positive reward to the agent if the selected action was in line with the difficulty threshold of the item estimated by the IRT model and the threshold chosen by the user. In this way, we wanted to favor items that had difficulties equal to or lower than the user's needs to reach a certain threshold, discouraging items that were too difficult to achieve the goal.

$$R_D = \begin{cases} \kappa_1 & \text{if } s_t \leq dt, \forall a \in A \\ \kappa_2 & \text{otherwise} \end{cases} \tag{1}$$

   In this function, if the action selected by the agent is in line with the IRT estimate and is less than or equal to the user's threshold, the agent will receive a positive reward κ_1. If not, the agent will receive a reward of zero κ_2.
   (b) $Actions$: For every action taken by the agent, this reward function was constructed to track the actions taken and to avoid items for which the

student has received a score equal to the highest possible from being presented again.

$$R_A = \begin{cases} \kappa_3 & \text{if } a_t \in actions\_used, \forall a \in A \end{cases} \quad (2)$$

where *actions_used* means the set of actions/items for which the student has already achieved the highest possible score. If the agent recommended an action in the *actions_used* it received a negative reward.

(c) *Episode*: The last reward function was related to the episode conclusion. Each episode was set to have a maximum duration related between (54%–150%) longer than the number of possible actions, to allow the agent to present the items again for which the subject had not reached the highest possible score and to reach the thresholds with the items with higher difficulty. If the agent could reach the established threshold within the maximum length of the episode, it received a positive reward; otherwise, it did not receive any reward.

$$R_E = \begin{cases} \kappa_4 & \text{if } s_t + s_{t+1} + ... + s_{t+n} \geq t \\ \kappa_5 & \text{otherwise} \end{cases} \quad (3)$$

At the end of each episode, the overall reward function was created based on the three functions. If the agent achieved a cumulative score on the items equal to or higher than the set threshold, then the reward function R included $R_D + R_A + R_E$. Otherwise, it only had $R_D + R_A$. R_D and R_A are considered intermediate rewards that should guide the agent in its choice of future actions.

4. Transition Probability Function P: It defines the probability of transitioning from one state to another after taking a specific action.

4.5 Hyperparameters

In this section, we summarized the hyperparameters used in each module. In Table 2, we have included the hyperparameters for configuring the reinforcement learning environment. Specifically, the PPO and A2C algorithms were trained for 10^5 timesteps across all modules, each for 1 h. The learning rate was set at 10^{-7}. Finally, the training algorithms were based on a policy object that implements an actor-critic approach, utilizing a 2-layer MLP with 64 units per layer [6]. It is true for some modules, while others utilize a custom network architecture.

Table 3 summarizes the hyperparameters associated with the custom environment, including the maximum length of each episode and its relationship with the number of possible actions. It also shows the number of thresholds considered in each module and the numeric values of the threshold (t) based on the estimated total score and the corresponding θ value (dt) obtained from the winning IRT solution. In addition, it considers N as the number of complete subjects' recordings for each module. This value has been extracted using the student's first attempt for each task in each module.

Table 2. Hyperparameters are implemented in both the PPO and A2C algorithms. lr: learning rate, ts: timesteps, Custom_net: Custom_network, policy: the policy implemented.

Module	RL configuration			
	policy	Custom_net	ts	lr
1	mlp	Yes: [128, 64]	10^5	10^{-7}
2	mlp	No	10^5	10^{-7}
3	mlp	No	10^5	10^{-7}
4	mlp	Yes: [128, 64]	10^5	10^{-7}
5	mlp	No	10^5	10^{-7}
6	mlp	Yes: [64, 32]	10^5	10^{-7}
7	mlp	No	10^5	10^{-7}
8	mlp	Yes: [64, 32]	10^5	10^{-7}
9	mlp	No	10^5	10^{-7}
10	mlp	Yes: [64, 32]	10^5	10^{-7}

4.6 Experiment Results

This study evaluated the performance of two reinforcement learning solutions, PPO and A2C, and a random baseline solution in collecting average rewards after 1000 episodes. The results, as illustrated in Fig. 1, demonstrate that the PPO solution outperformed both the A2C solution and the random baseline across all modules presented to subjects. A comparison of the mean improvement in collecting average cumulative rewards among the three solutions is summarized in Table 4. Evidently, the PPO solution achieved, on average, a 22.83% increase in rewards over the random action solution. Furthermore, this advantage in collecting rewards was consistent across all modules, with an improved range of 4.50% to 78.94% compared to the baseline. In contrast, the A2C algorithm demonstrated only a moderate improvement in collecting rewards, with an average increase of 1.29% over the baseline across all modules. This improvement was inconsistent, with a range of −8.99% to 7.69%.

5 Remarks and Discussion

The presented research centers on developing a recommender system that utilizes reinforcement learning and item response theory to enhance item recommendations for first-year bachelor's students in computer science taking a mathematics module. The integration of RL and IRT allows for personalized and adaptive recommendations based on the estimated difficulty threshold, enabling the system to suggest items within the user-selected threshold while avoiding items for which the student has already achieved the maximum possible score. In other words, the higher the threshold set by the student, the more complex the recommended items were, according to the θ value of the ETS distribution. This aspect

Table 3. Hyperparameters in the environment configuration. Length (%) is related to the maximum episode length and the relative percentage compared to the number of possible actions. n°t: is the number of thresholds included in the environment for that module. *dt*: is the difficulty threshold. *t*: is the threshold value. *N* is the number of complete subjects' recordings for each module.

Module	Environment configuration				
	length (%)	n°t	*dt*	*t*	*N*
1	20 (+82%)	4	$[-2.80, -1.96, .03, .57]$	[10.5, 15.5, 25.9, 28.3]	131
2	20 (+150%)	3	$[-1.05, 0, 2]$	[10.5, 23, 25]	129
3	8 (+100%)	2	[1.11, 2.56]	[10.5, 15.1]	132
4	20 (+150%)	3	$[-0.15, 0.63, 2.20]$	[10.5, 16, 26]	87
5	10 (+150%)	3	$[-0.15, 1.05, 1.60]$	[10.5, 15.4, 19.3]	99
6	10 (+100%)	3	[0.63, 1.05, 2.02]	[10.5, 13, 15]	100
7	20 (+150%)	3	$[-0.75, 0.33, 1.17]$	[10.5, 22.4, 32.2]	53
8	10 (+100%)	4	[18.7, 27, 38, 48]	$[-1.24, 0.03, 0.75, 3, 22]$	44
9	20 (+54%)	4	$[-2.68, -1, 0, 1]$	[10.5, 24.3, 33, 40]	50
10	25 (+79%)	4	$[-0.27, 0.33, 1.12, 2.62]$	[14.525, 17.31, 23, 31.5]	21

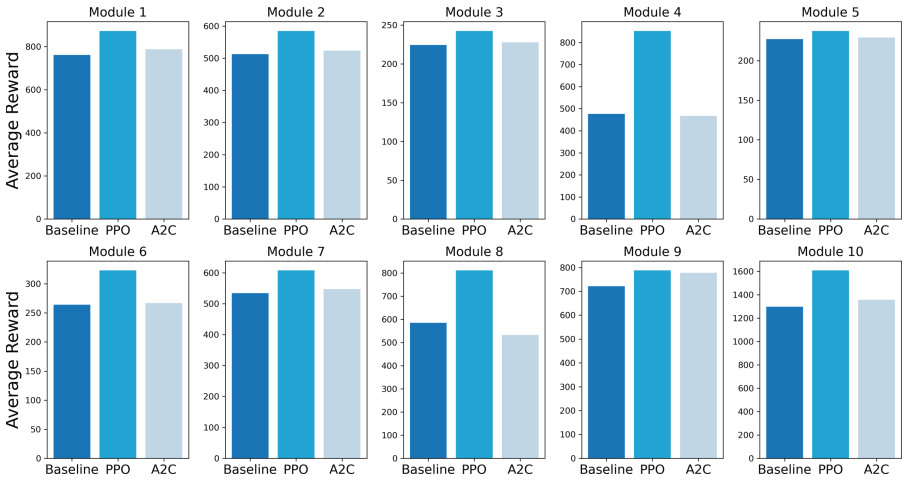

Fig. 1. Comparing the Performance of RL Algorithms and Baseline Across Modules. Average reward after 1000 episodes comparing Baseline, PPO, and A2C recommendations.

is particularly relevant because of the significance of allowing learners to determine their own difficulty level. As previously mentioned, interventions aimed at establishing personal academic goals are a crucial component in promoting student success. Moreover, by facilitating goal setting, students can develop a stronger sense of agency, intrinsic motivation, and self-directed learning skills.

Table 4. Performances comparison in the average reward between PPO and Baseline and A2C and Baseline actions. The values are expressed in percentual terms.

Module	PP0/Baseline	A2C/Baseline
1	+14.61	+3.44
2	+13.98	+2.08
3	+8.00	+1.47
4	+78.94	−1.87
5	+4.50	+0.9
6	+22.39	+1.1
7	+13.83	+2.53
8	+38.72	−8.99
9	+9.20	+7.69
10	+24.11	+1.27
Avg	+22.83	+1.29

The results demonstrate that incorporating RL solutions leads to improved performance, as measured by the average reward collected by the agents over 1000 episodes. Specifically, as highlighted in the results section, the PPO algorithm outperforms the A2C algorithm in every module, achieving an average reward that is 22.83% higher than the baseline.

Nevertheless, some considerations have to be mentioned. Firstly, while we have seen an improvement in the average reward collected, we need to determine if the recommendations benefit students. A future study should investigate this aspect more thoroughly. Secondly, our study used offline students' data for which we had complete answers for a module. It allowed us to use each episode as a new user and the answers as a transition over time for a specific user for that episode. This approach led to a policy strictly dependent on the answers collected, the students who answered all the items in each module, and the sample size and the possible transitions it learned. We only had a few dozen subjects for some modules who answered the entire set of items. In future studies, we plan to use this policy as a starting point and enhance it by incorporating online interaction between the user and the system. In addition, we used arbitrary thresholds derived from the estimated total score of the IRT solution, but there may be better options for achieving better results on test evaluations. In a future study, we plan to integrate this aspect by finding the best possible thresholds for each module that can provide the most informative guide for students to succeed on test evaluations. Lastly, we focused on item difficulty rather than the student's ability to solve a specific task. A future study should include this aspect in the RL environment to suggest items that also consider the student's ability to solve them.

6 Conclusion

This study presented a system for enhancing item recommendations for first-year bachelor's computer science students taking a mathematics module. The integration of Reinforcement Learning (RL) and Item Response Theory (IRT) allowed for personalized and adaptive recommendations based on the estimated difficulty threshold, enabling the system to suggest items within the user-selected scale while avoiding items for which the student has already achieved the maximum possible score. Results showed that incorporating RL solutions improved performance as measured by the average reward collected by the agents over 1000 episodes. Specifically, the proximal policy optimization algorithm outperformed the A2C algorithm in every module, achieving an average reward that is 22.83% higher than the baseline. Overall, this study provides valuable insight into the effectiveness of using IRT and RL for dynamic difficulty adjustment and the benefits of personalized task recommendation in educational settings. The proposed method can potentially improve learning outcomes and engagement in the domain of mathematics as well as other areas.

Acknowledgments. The authors would like to thank the German Federal Ministry of Education and Research (BMBF) for their kind support within the project *Personalisierte Kompetenzentwicklung und hybrides KI-Mentoring* (tech4compKI) under the project id 16DHB2208.

References

1. Arey, D., Wells, E.: Balancing act: «the art and science of dynamic difficulty adjustment». In: Game Developers Conference (2001)
2. Chalmers, R.P.: Mirt: a multidimensional item response theory package for the R environment. J. Stat. Softw. **48**, 1–29 (2012)
3. Constant, T., Levieux, G.: Dynamic difficulty adjustment impact on players' confidence. In: Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems, pp. 1–12 (2019)
4. Csikszentmihalyi, M.: Flow. The Psychology of Optimal Experience. New York (Harperperennial) (1990)
5. Embretson, S.E., Reise, S.P.: Item Response Theory. Psychology Press (2013)
6. Hill, A., et al.: Stable baselines3 (2020). https://github.com/DLR-RM/stable-baselines3
7. Hori, K., Fukuhara, H., Yamada, T.: Item response theory and its applications in educational measurement part i: item response theory and its implementation in R. WIREs Comput. Stat. **14**(2), e1531 (2022)
8. Leite, W.L., et al.: A novel video recommendation system for algebra: an effectiveness evaluation study. Association for Computing Machinery, New York (2022)
9. Lopes, J.C., Lopes, R.P.: A review of dynamic difficulty adjustment methods for serious games. In: Pereira, A.I., Košir, A., Fernandes, F.P., Pacheco, M.F., Teixeira, J.P., Lopes, R.P. (eds.) OL2A 2022. CCIS, vol. 1754, pp. 144–159. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-23236-7_11
10. Mnih, V., et al.: Asynchronous methods for deep reinforcement learning. In: International Conference on Machine Learning, pp. 1928–1937 (2016)

11. AI for Research: Student goal setting: an evidence-based practice student goal setting (2018)
12. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. In: International Conference on Learning Representations (2017)
13. Stelnicki, A.M., Nordstokke, D.W., Saklofske, D.H.: who is the successful university student? an analysis of personal resources. Can. J. High. Educ. **45**(2), 214–228 (2015)
14. Sutton, R.S., Barto, A.G.: Reinforcement learning: an introduction. J. Artif. Intell. Res. **4**, 1–53 (1998)
15. Tvarožek, J., Kravčík, M., Bieliková, M.: Towards computerized adaptive assessment based on structured tasks. In: Nejdl, W., Kay, J., Pu, P., Herder, E. (eds.) AH 2008. LNCS, vol. 5149, pp. 224–234. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-70987-9_25
16. Vygotsky, L.S., Cole, M.: Mind in Society: Development of Higher Psychological Processes. Harvard University Press (1978)
17. Xue, S., Wu, M., Kolen, J., Aghdaie, N., Zaman, K.A.: Dynamic difficulty adjustment for maximized engagement in digital games. In: Proceedings of the 26th International Conference on World Wide Web Companion, pp. 465–471 (2017)
18. Zhang, Y., Goh, W.: Personalized task difficulty adaptation based on reinforcement learning. User Model. User-Adap. Inter. **31**, 753–784 (2021)