Item Ordering Biases in Educational Data

Jaroslav Čechák and Radek Pelánek

Masaryk University Brno
Czech Republic
{xcechak1, pelanek}@fi.muni.cz

Abstract. Data collected in a learning system are biased by order in which students solve items. This bias makes data analysis difficult and when not properly addressed, it may lead to misleading conclusions. We provide clear illustrations of the problem using simulated data and discuss methods for analyzing the scope of the problem in real data from a learning system. We present the data collection problem as a variant of the explore-exploit tradeoff and analyze several algorithms for addressing this tradeoff.

1 Introduction

As students solve items in a learning system, the system can collect data on their performance. This data can be used to reason about student knowledge and learning and properties of items. Based on the results of data analysis we can take actions to improve student learning; the action can be either automatic (algorithmic) or manual (based on insight obtained from analysis) [1].

However, data from learning systems are biased by data collection, and these biases can lead to misleading conclusions [15]. A specific data collection issue is item ordering. Students often solve items in a similar order, specifically "from easier to more difficult". This ordering bias makes the analysis of data difficult as it confounds increase in item difficulty and student learning [4, 13, 15]. Moreover, data are often influenced by attrition bias—students leave a system in non-random order, e.g., due to mastery learning or self-selection bias. Samples of population solving different items are thus not representative. This issue has been studied previously in learning curves research [7, 10, 11].

For data analysis purposes it is beneficial to have some degree of randomization in the data collection process [12, 15]. In this way, we may overcome some of the biases. However, randomization usually goes against pedagogical principles, e.g., the progression "from easier to more difficult" is usually pedagogically desirable. We thus have a specific case of the explore-exploit tradeoff [2]: we want to present items to students in such a way that the collected data enable us to understand student learning and properties of items (*exploration*) and at the same time we want students to learn efficiently in the system (*exploitation*). The explore-exploit tradeoff in learning system has been in different variants studied in several recent works [3, 8, 9].

In this work, we focus specifically on biases related to item ordering. Although item ordering biases have been noted before, it was only as a sidenote of another analysis [4] or as one example of a wide discussion of biases [15]. We provide a deep analysis of the issue, which is relevant to many learning systems.

We use simulated data to provide a clear illustration of the item ordering bias and to discuss under what circumstances the problem is important and why. We also discuss methods for understanding data collected from real students (with respect to item ordering) and illustrate the application of these methods on real data from learning systems. Finally, we describe and compare several algorithms for addressing the explore-exploit tradeoff with respect to item ordering in learning systems.

2 Background

The basic setting for our analyses and discussions are problem solving exercises. These exercises are typically presented in an interactive form such as a web page or a mobile app. Students are presented with problem solving exercises with varying difficulty. The students may be working on the problem either in their free time or as a part of the class assignment. Examples of these problems include but are not limited to simple programming exercises, math problems, and logic puzzles.

The student performance is measured as the time required to solve the exercise. To be precise, we use a logarithm of time since problem solving times in learning systems are typically log-normally distributed [6]. While there are other possible performance measures, solving time is universally applicable to various exercise types.

2.1 Used Data

The analyses are performed on data collected from systems Umíme[1] and RoboMission[2]. There are four datasets in total, three from the former and one from the latter. These datasets cover three problem solving exercises: a logic puzzle with marble in a maze (*Marble* and *Marble2*), Sokoban logic puzzle (*Sokoban*) and block-based programming tasks (*RoboMission*).

These datasets contain student interaction logs where each entry represents an attempt of a student at solving an item (a single problem solving exercise). Each entry also holds information whether the student's attempt was successful or not. Overall statistics can be seen in Table 1. The distribution of the number of items seen by a given student and the number of students that have seen a given item is roughly geometric.

[1] https://www.umimeto.org/, available only in Czech
[2] https://en.robomise.cz/

Table 1. Basic statistics of the used datasets.

dataset	students	items	interactions
Marble	17868	110	258848
Marble2	2364	98	28810
Sokoban	12384	109	182979
RoboMission	4174	85	62534

Table 2. An overview of the simulation scenarios used in this paper.

name	skill distribution	learning	skill Δ	attrition
incremental	$\mathcal{N}(-3, 0.1)$	incremental	6	no
incr. small	$\mathcal{N}(-0.5, 0.1)$	incremental	1	no
step	$\mathcal{N}(-3, 0.1)$	step	6	no
steep	$\mathcal{N}(-3, 0.1)$	steep	6	no
diverse	$\mathcal{N}(-3, 3)$	incremental	6	no
diverse + att.	$\mathcal{N}(-3, 3)$	incremental	6	yes
const. + att.	$\mathcal{N}(0, 3)$	no	0	yes

2.2 Simulations

To better understand the behavior of the system and possible biases we used simulations. Simulations give us the possibility to control every aspect of the system and also provide us ground truth for observation comparison. The main function of our simulation framework can be simplified as take a student, while there is an available item, pick an item and generate solving time using a model. The source code of the framework is publicly available[3].

The student solving time model is defined as follows: $t = b + a\theta + \epsilon$. In this equation, t is a logarithm of solving time, b is an item difficulty, a is an item discrimination factor, θ is a student skill, and $\epsilon = \mathcal{N}(0, c^2)$ is Gaussian noise. In-depth description of the model is in [16]. To keep the simulation simple parameters a and c were fixed to values -1 and 1 respectively.

The framework is modular to allow modeling of different scenarios. They do not reflect reality but deliberately exaggerate some aspect to illustrate a specific bias. We choose to keep item difficulties fixed for better comparison. An overview of the scenarios used later in this paper can be found in Table 2.

The column *name* refers to a label we will use for the given scenario later in the paper. In the description of simulation results, we will also add a suffix indicating which item selection strategy was used.

Skill distribution describes how student skills were generated. We sampled the skill values from a normal distribution with parameters given in the table.

The column *learning* contains information about the type of student learning. Incremental means that a student is learning in small steps with each solved item. After each item, the student skill is increased by $\frac{\text{skill } \Delta}{\#\text{items}}$. Step learning

[3] https://github.com/adaptive-learning/simulations-aied2019

models a situation where students do not understand the concept, but they can "learn" the concept with some probability after solving an item. The learning happens by increasing the student skill by skill Δ with 5% probability after each solved item. The skill of a given student is increased at most once. We do not model knowledge deterioration, e.g., forgetting. Once the concept is mastered, the student keeps the gained skill. Steep learning is a situation where a student is learning rapidly and only a few items are enough to master the concept. Real life example would be learning of the user interface. In our simulation, it takes 10 items for a student to completely learn the concept.

The *skill* Δ states the maximal amount of skill the student can gain after solving every item. For example, if the student has a skill of -3.2 at the beginning and the skill Δ is 6. Then after solving all items, the student's skill will be 2.8.

The last column refers to whether the attrition is modeled or not. When there is no attrition, all students solve every item. When the attrition is modeled, all students have a fixed time budget they can spend solving items. For better illustration, it is set so that only a few students reach the last item. In our simulations, this happens to be at 0.6 times a sum of all items difficulties.

3 Item Ordering Bias

The data collected from student interactions in a learning system that uses a fixed item order may be biased. These biases manifest themselves as disproportions between the estimated item difficulty and the actual item difficulty. We use simulations to explore and clearly illustrate estimates in scenarios from Table 2. We also compare situations when students solve items from easier to more difficult (*ordered*) and in a random order (*random*).

Fig. 1 shows plots of different scenarios and their effect on estimated item difficulties. In this context, true item difficulty is equal to solving time of an average skilled student, i.e., the one with skill value of 0. The y-axis shows item difficulty. The x-axis corresponds to item order when sorted by their true difficulty. Curves for different scenarios show estimates of item difficulties computed as mean solving times. The translucent bands around them signify 95% confidence interval (computed by bootstrapping) for the mean.

Fig. 1 a) illustrates the importance of the relationship between student learning and item difficulty increase. We consider *incremental* and *incr. small.* scenarios in the *ordered* variant. In the *incremental* scenario, student learning dominates item difficulty increases, and the estimates of item difficulty are almost reversed, i.e., easier items are estimated as harder than truly hard items. In the *incr. small* scenario the item difficulty increases dominate. While easier items are still being overestimated and harder item underestimated, the relative ordering is reasonably well estimated.

Fig. 1 c) illustrates the results of the analysis under different assumptions about student learning. We keep the *incremental* scenario and compare it with *step* and *steep*. Students again solve items in the fixed order. The curve shapes differ substantially though in all three the difficulties of the first few items are

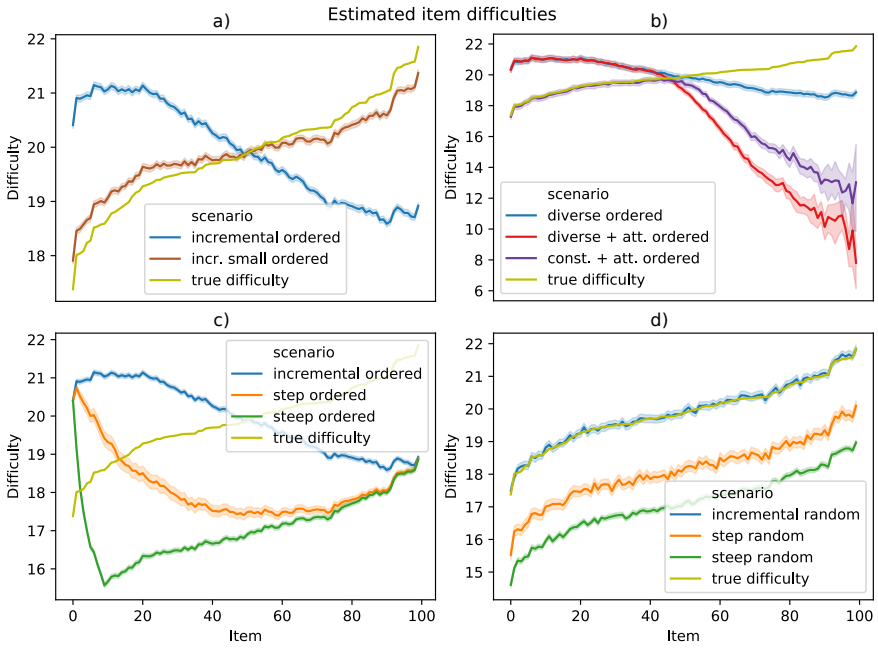

Fig. 1. Illustration of problem ordering bias. Note that in b) and d) the *true difficulty* (partially) overlaps with other curves.

highly overestimated. Real life example of this behavior could be introductory items teaching students how to use and work with the system.

Fig. 1 d) illustrates that the fixed item order causes problems with estimates. If we let students solve items in a random order, estimates become accurate in the sense of relative ordering. The absolute values of estimates and true difficulties differ as the expected value of student skill for a random student differs between the scenarios.

Fig. 1 b) illustrates the effect of attrition. In the *diverse* and *diverse + att.* scenarios, the student population is changing as students are learning in the process of solving items. In addition, the attrition leads to a change in a population sample for different items. Only the students with high skill are able to solve many items in a fixed amount of time. Consequently, later items are solved only by outstanding students. The similar result is achieved even in *const. + att.* scenario. In *diverse + att.* the learning amplifies this effect.

4 Understanding Data

Experiments with simulated data show that the fixed item order can have a significant impact on the analysis of data. To asses how relevant the problem is in a particular system, we need to understand the data. In what order do students solve items within a system? In some systems, the order is completely fixed. In

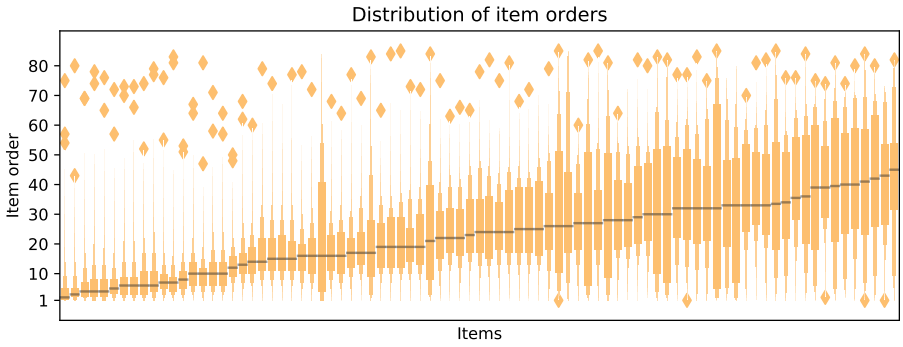

Fig. 2. Analysis of item ordering in RoboMission dataset.

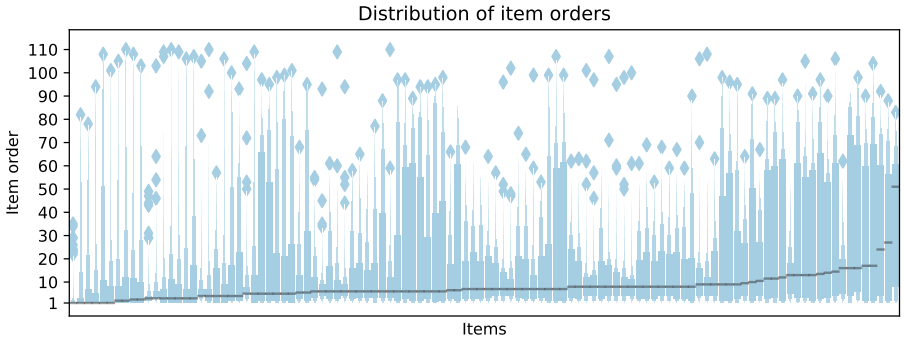

Fig. 3. Analysis of item ordering in Marble dataset.

such cases the potential bias caused by the ordering is high, but at least it is quite clear what is happening. In many realistic systems, however, the ordering of items is neither completely fixed nor random. One common setting is the following: items are presented to students in some specific default order, but students do not necessarily have to follow this order exactly. The provided ordering can be chosen manually or algorithmically; often it is explicitly or implicitly from easier items to more difficult.

To understand the data, we propose to use the following type of graph. For each student, we construct a student's item solving sequence by ordering the items the student has seen. The items are sorted from the first seen to the last seen. Aggregating over all students gives us a set of observed orders for any given item. We then visualize the distribution of these orders, sorting items by their median order.

An example of this visualization for the RoboMission dataset is in Fig. 2. The distributions for individual items are visualized using the "letter value" plot [5] (an extension of boxplot). Items with tall boxes usually have fewer observations (solutions). This visualization provides an overview of student behavior in the

system. In the specific case illustrated in Fig. 2, it is clear that although students do not proceed through the system in a completely identical order, there are strong regularities, particularly with respect to items solved at the beginning. Items in the RoboMission system are divided into levels and sublevels; in the graph, we can see "steps", where flat part are the items from the same sublevel.

For comparison, we include the same analysis for Marble dataset in Fig. 3. We can see that the median orders are slightly lower and they do not increase between items as much resulting in a "flatter" look. The former is caused by students solving on average fewer items resulting in shorter item sequences and lower item orders. The latter is caused by the randomness of the student item solving sequences. If orders of items are identically distributed on some finite interval, i.e., there is no precedence between items, the medians of all items are the same.

To better grasp the item ordering effect, it is useful to provide comparative analysis across different datasets and to quantify the effect. Fig. 4 presents analysis analogical to Fig. 2 and Fig. 3, but showing only the median order for each item and providing a comparison of several datasets. The plot shows the median of item orders (translucent) and corresponding linear regression lines (solid bold) for all datasets introduced earlier. The slope of the regression line indicates similarity of item sequences. The more students solve items in similar order the higher the slope.

For *Marble* and *Sokoban*, the median orders of all items are fairly low, and the slope is small as well. These are distinct exercises with the same recommendation behavior. In both a random unsolved item is recommended after the student had successfully finished solving the item. The *Marble2* dataset is from the same exercise as *Marble* with a change in the recommender behavior. The system recommends items in a fixed order created by an expert. We can see a substantial increase in the slope. In *RoboMission* the items are divided into nine levels and each level is further divided into three sublevels. The recommended item is always taken randomly from the easiest not yet mastered sublevel. Hence the recommendations are ordered in terms of sublevels while random within the sublevel. The slope is between a fixed and random ordering as one would expect from the semi-ordered items. Note that similar type of analysis (using slightly different visualization) was used by [4].

In all four cases, the students may ignore the recommendation and choose any item from an item overview. Moreover, students typically solve only a few items. Consequently, most median item orders are low. The items in the overview are presented in some kind of lattice resulting in an implicit ordering that some students may choose to follow. Due to these factors, the extreme values of the slope are unlikely in real life scenarios.

Slope lines may not intersect the origin as evident from the Fig. 4. The slope intersecting x-axis to the right of the origin indicates that students choose items randomly in the beginning and later begin to follow a common trend. The slope intersecting on the left-hand side of the origin indicates that students solve more randomly chosen items later as they progress through the system.

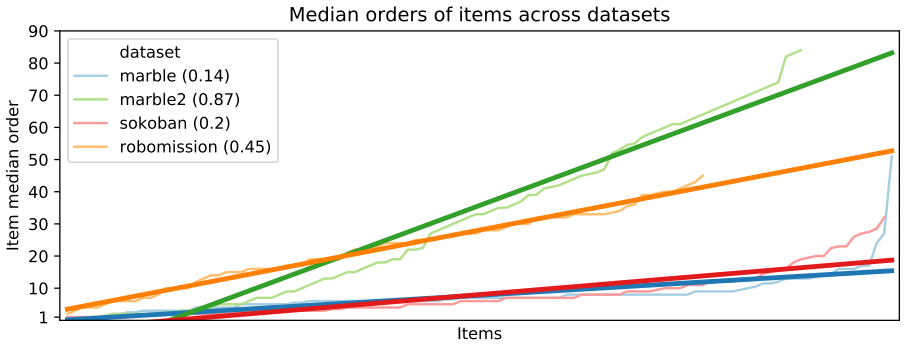

Fig. 4. Analysis of ordering from real systems. The regression line slopes for each dataset are located in parentheses in the plot legend.

One may choose to interpolate the item medians with polynomials or different kinds of curves. This is a viable option, and by definition, the fit will be tighter. Linear regression has the benefit of being non-parametric. Fig. 4 illustrates that even the simple slope analysis can differentiate between various system behaviors. It could, in turn, be used to asses the relevance of the item ordering related biases in a system.

5 Algorithms for Dynamic Item Ordering

In learning systems, we would like the items to be ordered well for efficient student learning, i.e., from easier to more difficult. We call it the ideal ordering. At the same time though, we would like "good" orderings for the analysis. As evident from the earlier simulations that would be a randomized ordering. We can perceive the situation as a variation on exploit-explore tradeoff. The problem lies in balancing exploitation (students are learning efficiently) and exploration (obtaining unbiased data about items).

It would be possible to avoid the use of randomization and to fully focus on exploitation if we take student learning into account in the analysis. The shown bias arises from student learning, and the modeling of student learning removes the bias. However, this approach is valid only when the used model of student learning fits very well the actual progress of student learning. There are many different student modeling approaches [14], and it is nontrivial to choose a suitable model for a particular situation. Moreover, all models are simplifications, and even the use of an appropriate type of model can cause distortions in model analysis. It thus makes sense to explore "model-free" methods for the analysis.

5.1 Algorithms

We have devised three algorithms that combine exploration with exploitation in various ways. The algorithms are quite simple and definitely could be improved

upon. Our main aim is to illustrate the methodology that could be used to compare such algorithms.

First K random The algorithm lets the first K students go through the items in a completely random order. After the Kth student, the estimates for item difficulty are computed for each item as a mean problem solving time for the K students. The items are sorted based on their estimates from the easiest to the most difficult. This order is then fixed for any later student.

ϵ-greedy The algorithm keeps an approximation of the ideal ordering. With probability 1-ϵ a student follows this ordering. With probability ϵ a student gets randomly ordered items. The approximation could initially be set to any ordering desired, in our case some random ordering. The item difficulty estimates are updated after each student that got the randomly ordered items. Only times of students with random passes are used for estimates.

Adaptive periodic K The algorithm keeps an ordering based on estimates of item difficulties. The initial ordering is random, and the estimates are recomputed after K students have gone through the system since the last update. Times of all the previous students are used in the computation of estimates.

5.2 Experiments

We once again used the simulations to get insight into the properties of these algorithms in different scenarios. We are mainly interested in convergence properties, i.e., can the algorithm find the ideal ordering. The metric we choose is the Spearman's correlation coefficient with ideal ordering. The Fig. 5 shows the progression of the correlation as new students are coming to the system. Each data point is a mean of correlations for ten consecutive students. This step is done to smooth out the curves yet still retain some local variations.

In the case Fig. 5 a) we can see that all three algorithms are performing well. This is the scenario where problem solving time estimates follow true item difficulties. Fig. 5 b) shows results for a scenario where the ordering bias is highly present. We can see that *Adaptive periodic 20* algorithm is struggling and oscillating around a non-ideal ordering. Even in this scenario, 20 random student passes are enough to obtain a reasonable approximation of the ideal ordering as illustrated by *First 20 random* algorithm. The scenario c) adds attrition on top. We can see that the performance of all algorithms is impacted. The *0.05-greedy* algorithm is still able to slowly approach the ideal ordering. We can compare the scenario d) to others to gain some insight into the interplay of attrition and learning. Compared to c) we can see that all algorithms are performing better. *Adaptive periodic 20* is doing even better than in scenario b) with learning and no attrition. Other algorithms are affected by limited information gained from student solving only a subset of items.

It is worth noting that the robustness of *ϵ-greedy* algorithm is achieved at the expense of some students having randomly ordered items. That is also the reason for the ruggedness of its curve in the plots.

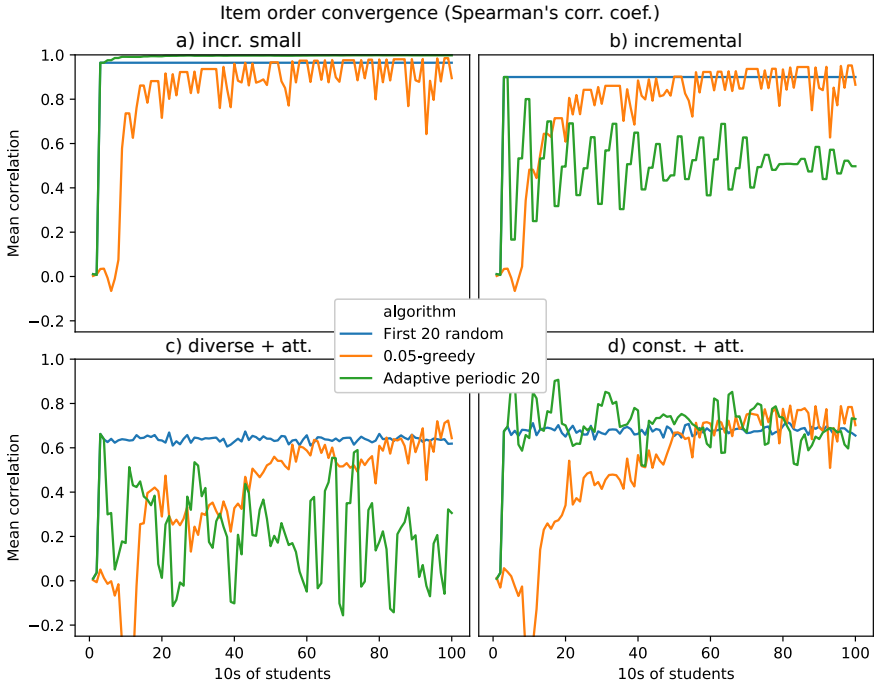

Fig. 5. Convergence of ordering computed as the Spearman's correlation coefficient.

6 Discussion

Students often solve items in a specific order, particularly in a progression from easier to more difficult. Such regularities in student behavior lead to biases in data collected from learning systems. When not taken into account in the analysis, these biases may lead to misleading conclusions. We have illustrated the potential biases using simulated data, showing how the importance of biases depends on the speed and nature of student learning (relative to the increases in item difficulty). We have also proposed a technique for analyzing the degree of ordering bias in data from real systems.

To overcome biases caused by fixed ordering, we can utilize dynamic item orderings. An interesting research question is how to realize such a dynamic ordering. In this work, we analyzed three simple algorithms for a dynamic ordering of items. Using simulated data we showed that their performance depends on the specific application scenario (what is the speed and nature of student learning, what is the distribution of item difficulties). The results show that under some settings, naive use of automatic adaptation can lead to worse results than solutions based on simple randomization. An interesting direction for future work is to consider more complex combinations of adaptation and randomization.

References

1. Aleven, V., McLaughlin, E.A., Glenn, R.A., Koedinger, K.R.: Handbook of research on learning and instruction, chap. Instruction based on adaptive learning technologies. Routledge (2016)
2. Audibert, J.Y., Munos, R., Szepesvári, C.: Exploration–exploitation tradeoff using variance estimates in multi-armed bandits. Theoretical Computer Science **410**(19), 1876–1902 (2009)
3. Clement, B., Roy, D., Oudeyer, P.Y., Lopes, M.: Multi-armed bandits for intelligent tutoring systems. Journal of Educational Data Mining **7**(2), 20–48 (2015)
4. González-Brenes, J., Huang, Y., Brusilovsky, P.: General features in knowledge tracing: Applications to multiple subskills, temporal item response theory, and expert knowledge. In: Proc. of Educational Data Mining. pp. 84–91 (2014)
5. Hofmann, H., Wickham, H., Kafadar, K.: Letter-value plots: Boxplots for large data. Journal of Computational and Graphical Statistics **26**(3), 469–477 (2017)
6. Jarušek, P., Pelánek, R.: Analysis of a simple model of problem solving times. In: Proc. of Intelligent Tutoring Systems. LNCS, vol. 7315, pp. 379–388. Springer (2012)
7. Käser, T., Koedinger, K.R., Gross, M.: Different parameters - same prediction: An analysis of learning curves. In: Proc. of Educational Data Mining. pp. 52–59 (2014)
8. Lan, A.S., Baraniuk, R.G.: A contextual bandits framework for personalized learning action selection. In: Proc. of Educational Data Mining. pp. 424–429 (2016)
9. Liu, Y.E., Mandel, T., Brunskill, E., Popovic, Z.: Trading off scientific knowledge and user learning with multi-armed bandits. In: Proc. of Educational Data Mining. pp. 161–168 (2014)
10. Murray, R.C., Ritter, S., Nixon, T., Schwiebert, R., Hausmann, R.G., Towle, B., Fancsali, S.E., Vuong, A.: Revealing the learning in learning curves. In: Proc. of Artificial Intelligence in Education. pp. 473–482. Springer (2013)
11. Nixon, T., Fancsali, S., Ritter, S.: The complex dynamics of aggregate learning curves. In: Proc. of Educational Data Mining (2013)
12. Papoušek, J., Stanislav, V., Pelánek, R.: Evaluation of an adaptive practice system for learning geography facts. In: Gasevic, D., Lynch, G., Dawson, S., Drachsler, H., Rosé, C.P. (eds.) Proc. of Learning Analytics & Knowledge. pp. 40–47. ACM (2016)
13. Pavlik, P.I., Yudelson, M., Koedinger, K.R.: A measurement model of microgenetic transfer for improving instructional outcomes. International Journal of Artificial Intelligence in Education **25**(3), 346–379 (2015)
14. Pelánek, R.: Bayesian knowledge tracing, logistic models, and beyond: an overview of learner modeling techniques. User Modeling and User-Adapted Interaction **27**(3), 313–350 (2017)
15. Pelánek, R.: The details matter: methodological nuances in the evaluation of student models. User Modeling and User-Adapted Interaction (2018)
16. Pelánek, R., Jarušek, P.: Student modeling based on problem solving times. International Journal of Artificial Intelligence in Education **25**(4), 493–519 (2015)