Adaptive Test Recommendation for Mastery Learning

Nassim Bouarour
CNRS, Univ. Grenoble Alpes
Grenoble, France
nassim.bouarour@univ-grenoble-alpes.fr

Idir Benouaret
Epita Research Laboratory
Lyon, France
idir.benouaret@epita.fr

Cédric d'Ham
CNRS, Univ. Grenoble Alpes
Grenoble, France
cedric.dham@univ-grenoble-alpes.fr

Sihem Amer-Yahia
CNRS, Univ. Grenoble Alpes
Grenoble, France
sihem.amer-yahia@univ-grenoble-alpes.fr

ABSTRACT

We tackle the problem of recommending tests to learners to achieve upskilling. Our work is grounded in two learning theories: mastery learning, an instructional strategy that guides learners by providing them tests of increasing difficulty, reviewing their test results, and iterating until they reach a level of mastery; Flow Theory, which identifies different test zones, frustration, learnable, flow and boredom zones, to determine the best k tests to recommend to a learner. We formalize the AdUp Problem and develop a multi-objective optimization solution that adapts the difficulty of recommended tests to the learner's predicted performance, aptitude, and skill gap. We leverage existing models to simulate learner behavior and run experiments to demonstrate that our formalization is best to attain skill mastery. We discuss open research directions including the applicability of reinforcement learning and the recommendation of peers in collaborative projects.

ACM Reference Format:
Nassim Bouarour, Idir Benouaret, Cédric d'Ham, and Sihem Amer-Yahia. 2023. Adaptive Test Recommendation for Mastery Learning. In *2nd International Workshop on Data Systems Education: Bridging education practice with education research (DataEd '23), June 23, 2023, Seattle, WA, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3596673.3596977

1 INTRODUCTION

The rapid growth in new learning opportunities e.g., MOOCs, tutorials, and community-based discussion forums, is shifting attention to online skill improvement. Upskilling that is occurring outside of formal offerings is a fast-growing segment of the educational economy [?]. Yet, there is little algorithmic work that focuses on crafting dedicated strategies to reach high skill mastery. To the best of our knowledge, our work is the first to propose a formalization and develop algorithms for skill mastery.

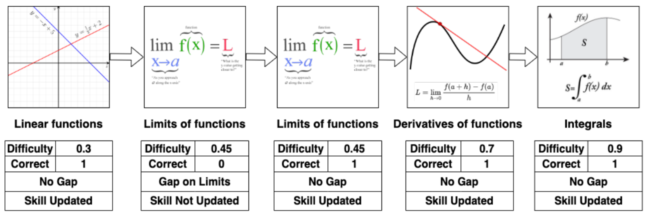
$y=-x+5$
$y=\frac{1}{2}x+2$
$\lim_{x \to a} f(x) = L$
function
"What is the y-value getting closer to?"
"As you approach a along the x-axis"
$L = \lim_{h \to 0} \frac{f(a+h)-f(a)}{h}$
$f(x)$
S
a
b
$S=\int_a^b f(x)\,dx$
Linear functions
Limits of functions
Limits of functions
Derivatives of functions
Integrals
Difficulty 0.3 | Correct 1 | No Gap | Skill Updated
Difficulty 0.45 | Correct 0 | Gap on Limits | Skill Not Updated
Difficulty 0.45 | Correct 1 | No Gap | Skill Updated
Difficulty 0.7 | Correct 1 | No Gap | Skill Updated
Difficulty 0.9 | Correct 1 | No Gap | Skill Updated

Figure 1: Example of the process of learning Math.

Today, learners engage in self-directed learning, managing many elements of their own study, which, in turn, often requires working on various learning activities independently with less direct guidance from teachers [?]. Consequently, providing guarantees on the quality of learning outcomes is increasingly difficult in these new bite-sized learning structures as they can lead to the so-called illusion of explanatory depth [?] where learners only acquire a superficial understanding of a topic. Ideally, each learner should receive tests chosen in a such way that the learner's skill progresses. This should account for the learner's ability to resolve tests based on skill and past performance. That is the topic of mastery learning [?] where the focus of instruction is the time required for different learners to acquire the same competencies and achieve the same level of mastery. This is very much in contrast with classic models of teaching where all learners are given approximately the same amount of time to learn. We illustrate that with an example.

Motivating example. Consider a learner with very basic math knowledge who wants to learn mathematical functions. Figure 1 illustrates an example of the learning process. In the beginning, the learner receives tests with a moderate difficulty level of 0.3 for which they provide correct answers. As a result, they incur no skill gap, and their skill is estimated. This triggers a second step where they are assigned more difficult tests (on limits of functions) for which they fail. In addition to not increasing their skill, they incur a skill gap. To fill that gap, they are given a second chance with the same type of tests in which they succeeded. Their input is correct and their skill increases. The same process is repeated, and the learner receives more difficult tests on derivatives and then on integrals. They provide correct results and their skill increases.

Challenges. Our example identifies several challenges. First, we need to determine which k tests to assign to a learner at each iteration. Existing work on recommending tests optimizes the learner's

DataEd '23, June 23, 2023, Seattle, WA, USA
© 2023 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-0207-5/23/06…$15.00
https://doi.org/10.1145/3596673.3596977

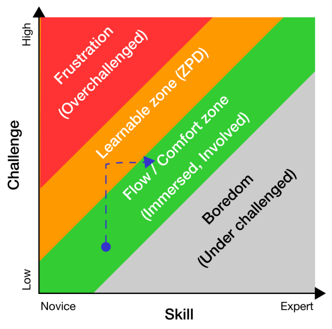

Figure 2: Illustration of the combination of the Zone of Proximal Development (ZPD) [?] and Flow Theories [?]. In [?], it is shown that learners improve their skills by completing tests that are more but not too challenging (dotted line).

expected performance either by assuming tests with the same difficulty level [?] or by pre-defining the composition of difficulties beforehand (e.g., by alternating test difficulty levels [?]). Indeed, according to learning theories illustrated in Figure 2, simply relying on the learner's expected performance runs the risk of narrowing down the learner into a zone of "boring" and under challenging tests that do not incur upskilling. To address that, we propose to also account for the learner's aptitude, i.e., the difference between the learner's skill and the test difficulty level. This will encourage selecting tests that challenge the learner (the learnable zone in Figure 2). Hence, we need to balance expected performance and aptitude. Second, we need to account for the skill gap in determining the next k tests. To the best of our knowledge, no existing work does so. Third, we need to simulate the learners' performance and devise a skill update strategy after they complete a batch of k tests.

Contributions. We formalize the AdUp Problem, our Adaptive Upskilling Problem as an optimization problem where a learner receives k tests that maximize expected performance and aptitude, and minimize accumulated skill gap. The combination of these objectives constitutes the novelty of our formalization. Most related work focuses on modeling learners and neglects the test selection question [?]. We adopt a Pareto solution by defining dominance between k test sets and develop a heuristic algorithm based on *Hill Climbing* [?] that finds a subset of the solutions [?]. We use real data collected from a Czech educational system MatMat (matmat.cz)[1] to infer test difficulty levels. To simulate learners and predict their probability of providing correct answers, we leverage an extended version of *Bayesian Knowledge Tracing (BKT)* [?] that leverages test difficulties [?]. After each iteration, the skill of a learner is updated following existing approaches that aggregate consecutive correct answers [?].

Our empirical study examines the impact of the optimization dimensions on upskilling and mastery. We run a simulation for 100 learners. We compare our solution to different variants (optimizing single or two objectives, alternating difficulty levels). We find that optimizing one objective is worse than optimizing two or three objectives. We confirm that alternating skill difficulty yields good skill gain and progression but is not well-adapted to attain skill mastery. We verify that optimizing aptitude is required to attain mastery and that combining aptitude, expected performance, and skill gap reduce the number of iterations needed to attain mastery.

Implications for education systems and databases. Our work is closely related to the domain of education data systems. Our framework can constitute a solution for different challenges of AI-based assessments [?]. It could be adapted to either design an adaptive educational system [?] (e.g., Desire2Learn [2]) or be applied directly to extend one of the existing systems: e.g., the LabNbook [3] environment at our university that scaffolds students' activity as they learn to write experimental protocols. It also aims to assist instructors as it might be challenging for them to provide rich and timely support to students [?]. It aims to keep instructors in the decision loop [?] and not replace them, by either giving them the chance to develop the content of tests or assisting them in choosing the best tests to assign. Also, our work might propose a starting solution to address AI explainability in education [?] as it optimizes interpretable optimization-related dimensions for test assignments. Our work is also relevant to teaching databases as described in our recent publication [?]. It could be used to extend the constraint-based tool SQL-Tutor [?] to produce a hybrid tutoring paradigm to help students master writing SQL queries.

2 MODEL AND PROBLEM

We consider a learner $l \in \mathcal{L}$ who follows an iterative learning process for a given skill sk. We assume one and only one skill in this work that has a scalar as a value. Extending the skill representation to a vector is not straightforward. It requires studying independence between skills or making an independence assumption which would be unrealistic in most scenarios.

At each step, l completes a set of k tests with different difficulty levels for sk. Each test $t \in \mathcal{T}$ has a skill difficulty d_t that remains unchanged. We associate to each learner l a skill value $l.sk$ that either remains the same or increases as the learner successfully completes tests. The initial value of $l.sk$ can be computed from the information the learner fills when joining the system (e.g., by completing an initial set of tests or through a pre-assessment questionnaire).

We aim to formalize a problem where at any given iteration, the learner receives a batch of k tests whose difficulty level is greater than $l.sk$. To define our problem, we formalize dimensions that characterize the learning process of a learner l for a skill sk.

2.1 Expected performance, aptitude, and gap

Expected performance. It is the expected performance of learner l for a test t. It is based on the similarity of t with successfully completed tests $l.\mathcal{S} \subseteq \mathcal{T}$ by l and is formalized as follows:

$$exPerf(l, t) = sim(t, l.\mathcal{S})$$

Aptitude. It quantifies the difference between a learner's skill value ($l.sk$) and the difficulty level of a test t (d_t). It represents the learner's progression ability for the skill when assigned tests that are correctly completed. Aptitude is defined as follows:

$$apt(l, t) = d_t - l.sk$$

[1]https://github.com/adaptive-learning/matmat-web/blob/master/data/data_description.md

[2]https://www.d2l.com/

[3]https://labnbook.fr/en

Gap. It quantifies the distance between the past failed tests of learner l (set $l.\mathcal{F} \subseteq \mathcal{T}$) and the test t and is defined as follows:

$$gap(l, t) = dist(t, l.\mathcal{F})$$

Similarity and distance between tests can be computed in several ways. In our implementation, we use the Euclidean distance between the difficulty levels of tests.

2.2 The AdUp problem

To achieve skill mastery, we propose an iterative formulation that solves the following problem:

Problem 1 (The AdUp Problem). *Given a learner l, with a skill $l.sk$, find a batch $B \subseteq \mathcal{T}$ of k tests to assign to l at iteration i s.t.:*

$$\begin{aligned} &\textit{maximize} \sum_{t \in B} exPerf(l, t) \\ &\textit{maximize} \sum_{t \in B} apt(l, t) \\ &\textit{minimize} \sum_{t \in B} gap(l, t) \\ &\textit{subject to } \ |B| = k \end{aligned} \tag{1}$$

The main challenge in solving the AdUp problem, is the multi-objective nature of the problem. Naive solutions would be, using a weighted sum method by transforming the problem into a single objective optimization, or ϵ-Constraint method where a single objective is optimized and the remaining ones are restricted within user-specific values [?]. These methods suffer from the need to fix weights or the thresholds of the restricted objectives and cannot provide optimal solutions when the objectives are conflicting.

3 ALGORITHM

We propose a method that finds the Pareto solutions by addressing all objectives at once [?]. To do so, we define a dominance relation between two sets of size k.

We represent the set of all test batches as $C_k = \{B | B \in \mathcal{T}, |B| = k\}$. We define batch dominance $B_1 \succ B_2$ between any two sets in C_k:

Batch dominance. We say that B_1 dominates B_2 ($B_1 \succ B_2$) iff:

- B_1 is no worse than B_2 for all three objectives.
- B_1 is strictly better than B_2 for at least one objective.

We design a heuristic Algorithm 1 to avoid an exhaustive exploration of the whole search space. It starts by performing *times* iterations where in each it finds an optimal batch of tests (Lines 3 to 7) to avoid local optimums. At each iteration, it first generates a random candidate. Then it performs *Hill Climbing* to optimize both expected performance and aptitude. The returned candidates are added to the set of results. From this set, only non-dominated candidates are kept (Line 8). Finally, the candidate that yields the lowest skill gap is chosen (Line 9) and assigned to the learner (Line 10). The learner's skill is updated after the completion of the test batch (Line 11). Refer to Section 4.3 for our skill update strategy. This process is repeated until the learner l achieves skill mastery.

Algorithm 2 searches over all the neighbors of the input batch and selects the one that improves aptitude and expected performance. A neighbor of a batch is computed by replacing one and only one test with another test that has either the next higher or lower difficulty (Lines 3 to 10). If all neighbors are dominated by the current batch, this latter is chosen as the optimized batch. Otherwise, the algorithm replaces the current batch by randomly selecting one from the non-dominated neighbors.

Algorithm 1: Heuristic MOO

```
   Input: learner $l$, set of tests $\mathcal{T}$, size $k$, # repetition $times$
1  while not mastery do
2      $Results \leftarrow \emptyset$
3      for $n$ in $[1..times]$ do
4          $C \leftarrow Random\_candidate(k)$
5          $C^* \leftarrow HCAE(C)$
6          $Results.Add(C^*)$
7      end
8      Keep non-dominated candidates in $Results$
9      $B \leftarrow$ The solution from $Results$ with the lowest skill $gap$
10     $l$ completes $B$
11     $l.sk \leftarrow skill\_update(l.sk, B)$
12 end
```

Algorithm 2: HCAE - Hill Climbing for Aptitude and Expected Performance

```
   Input: Batch of $k$ tests $B$
   Output: Optimized batch $B^*$
1  while True do
2      $Candidates \leftarrow \emptyset$
3      for $test \in B$ do
4          $test\_down \leftarrow$ A test with the next lower difficulty
5          $B\_1 \leftarrow B - \{test\} + \{test\_down\}$
6          $test\_up \leftarrow$ A test with the next higher difficulty
7          $B\_2 \leftarrow B - \{test\} + \{test\_up\}$
8          $Candidates.add([B\_1, apt(B\_1), exPerf(B\_1)])$
9          $Candidates.add([B\_2, apt(B\_2), exPerf(B\_2)])$
10     end
11     Keep non-dominated candidates in $Candidates$
12     if $B$ dominates all candidates in $Candidates$ then
13         return $B$
14     end
15     else
16         $B \leftarrow$ A random candidate from $Candidates$
17     end
18 end
```

4 EXPERIMENTS

The purpose of our experiments is to verify the assumptions we made in this work, namely (i) alternating skill difficulty yields a good skill gain and progression but is not well-adapted to attain skill mastery, (ii) optimizing aptitude is required to attain mastery, and (iii) optimizing all three dimensions reduces the number of iterations needed to attain mastery.

4.1 Data

We use real data collected from a Czech educational system[4]. It is an adaptive practice system for elementary arithmetic tests. The data contains more than 1800 tests from which we infer 42 distinct difficulty levels ranging in $]0, 1[$. We assume this order of difficulty level: "divisions" > "multiplications" > "subtractions" > "additions" > "numbers". We also assume multi-digit operations are more difficult than single-digit ones. We also use the display type of the tests as a feature to infer difficulty. We consider that tests displayed with visual examples are simpler than written tests. We assume that all tests for "numbers" have the lowest difficulty (0.13). The difficulty ranges of "additions", "subtractions", "multiplications", and "divisions" are $[0.2, 0.4[$, $[0.4, 0.6[$, $[0.6, 0.8[$, and $[0.8, 1[$ respectively. For each type of operation, and starting from the beginning of its range, we increase the difficulty with a fixed rate of tests that encounter multi-digit numbers and tests displayed without any example.

4.2 Learner simulation

We simulate learners using an extended version of BKT (KT-IDEM) as it is a cognitively diagnostic form of assessment that has been recognized as beneficial to learners and instructors [?]. BKT models the learning process given the chronological sequence and correctness of tests. It infers the knowledge of learners by predicting the probability of learning. In addition to this inferred probability, two more probabilities are used to estimate the performance of the learner: Guess and Slip probabilities. Guess is the probability of correctly answering a test when the learner does not master the difficulty while Slip is the probability of incorrectly answering a test even if the learner masters the difficulty. If the test is easy, the probability of Guess is high. If the test is hard, the probability of Slip is high as the learners are likely to make mistakes [?].

4.3 Skill update

After the completion of a batch B of k tests, we update the skill of learner l as follows:

$$skill\_update(l.sk, B) = max_{sk \in D \cup \{l.sk\}} sk \qquad (2)$$

where D is the set of difficulty values of correctly completed tests for which all tests with lower difficulties were correctly completed.

To show the intuition of this strategy, we consider a learner with $l.sk = 0.3$ at iteration i. At the next iteration $i + 1$, the learner is targeted with $k = 3$ tests t_4, t_5, and t_6 having 0.35, 0.4, and 0.45 as difficulty levels respectively. We consider that the learner correctly answered t_4 and t_6 and failed t_5. Using our strategy, the skill value $l.sk$ is updated to be equal to 0.35 (difficulty of t_4). The correct completion of t_6 is not considered as exists one test (t_5) with a lower difficulty that was wrongly completed. To account for variability in learners, we used the static mastery detection method *NCC* [?] that updates the skill if the number of consecutive correct answers, for a given difficulty level is N.

4.4 Variants

As we aim to study the impact of each optimization dimension, we devise several variants that produce a batch of k tests: MOO: uses the multi-objective optimization Algorithm 1. MOEG, MOAG, and MOAE: optimize expected performance and gap, aptitude and gap, or aptitude and expected performance respectively. MOG, MOE, and MOA: optimize gap only, expected performance only, or aptitude only. ALTERNATE: assigns a random set of k tests whose difficulty levels alternate in round-robin: k easy then k medium then k hard [?].

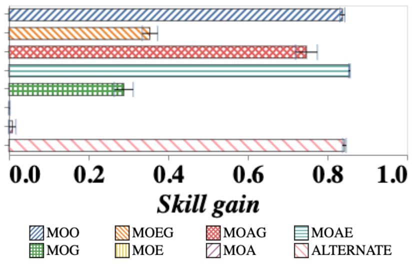

Figure 3: Average skill gain for each variant.

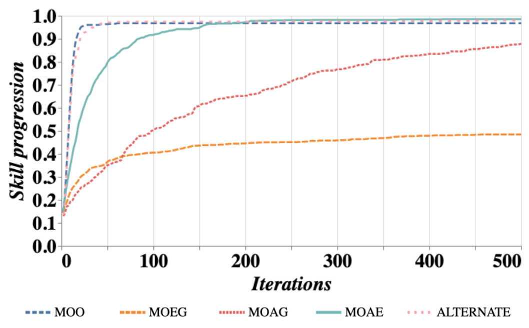

Figure 4: Skill progression as a function of # iterations.

We assume learners attain mastery when their skill value equals the highest difficulty level. We set the maximum number of iterations to 500. We vary the value of k in $\{3, 5, 10, 15, 20\}$ and the number of simulations, i.e., learners, in $\{50, 100, 300\}$. Due to lack of space, we only report results of 100 simulations for $N = 1$, $k = 3$. For other settings, we see similar observations and we refer the reader to our GitHub repository[5] for our complete results and code.

4.5 Simulation results

We first report (1) the average skill gain and (2) the average skill progression in each variant. To better understand this first experiment, we examine (3) the percentage of learners who attained mastery and (4) the average number of iterations required to attain mastery. Finally, we compute (5) the average time each variant takes to generate a batch of k tests.

Skill gain and progression. Figure 3 reports the difference between the last and first skill values for all simulated learners (average skill gain). We observe that MOO and MOAE produce the highest average skill gain. Surprisingly, ALTERNATE seems to also produce a high skill gain. To elucidate that, we plot Figure 4 to examine the average step-wise skill progression. Here again, we observe that MOO and MOAE result in the fastest upskilling with a clear advantage for the former. MOAG is slower than these two variants but still

[4]https://github.com/adaptive-learning/matmat-web/blob/master/data/data_description.md

[5]https://github.com/AdaptiveUpskilling/AdUp.git

quicker than MOEG. This reinforces our initial assumption that optimizing for all three objectives at once yields the best results. This experiment also shows that ALTERNATE yields a high skill progression. This confirms our initial assumption that existing alternating task difficulties do yield good skill gain and progression. Next, we examine whether ALTERNATE compares favorably to MOO and MOAE in terms of achieving skill mastery.

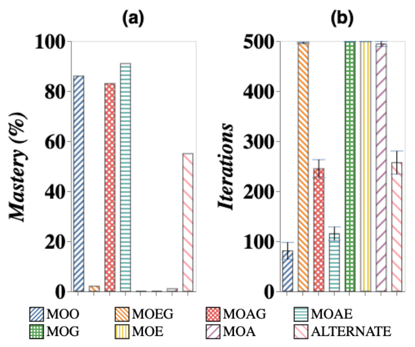

Figure 5: (a) Percentage of learners who attain mastery - (b) Average number of iterations to attain mastery.

Mastery. Figure 5 (a) reports the number of times each variant attained mastery. One can see that while ALTERNATE reaches a reasonable mastery level (≈ 59%), it is much lower than MOO, MOAG and MOAE (≈ 90%). This clearly confirms that aptitude plays a central role in attaining mastery. Hence, while alternating test difficulty levels in ALTERNATE does achieve good skill gain and skill progression performances, it is capped in terms of mastery level since it does not explicitly optimize aptitude. We can also observe that single-objective variants rarely attain mastery. This experiment confirms our initial assumptions: MOE assigns tests that are similar to the ones the learner completed correctly, thereby staying within the under-challenging zone [?]. MOA assigns tests that are too difficult and that keep the learner in a frustration zone [?].

Figure 5 (b) shows the average number of iterations to attain mastery for each variant. One can observe that ALTERNATE attains mastery in a similar number of iterations as MOAG but has a lower rate of mastery. Nevertheless, it is quicker than all single-objective variants. As explained before, these variants narrow the learners into zones where their skill value does not evolve while ALTERNATE offers more challenging batches which allow learners to attain mastery more often. However, simulated learners under ALTERNATE are able to correctly complete difficult tests but are unable to do so for the most difficult tests. While MOAE attains a slightly higher mastery level than MOO, it is outperformed by MOO in terms of the number of iterations needed to achieve mastery.

Response time. MOO has the worst time average as it has to optimize three objectives (≈10 seconds). MOAE would be a good candidate since it runs faster than MOO. However, MOO does better than MOAE on skill progression and on the average number of iterations needed to attain mastery. Therefore, we will need to focus on improving response time for MOO in future work.

5 RELATED WORK

Education Science. Flow [?] and ZPD [?] theories conceptualize the idea of *experiential learning* [?] that emphasizes the importance of choosing appropriate tests for learners. Flow theory was shown to be effective in the physical world in on-the-job training [? ?]. More recently, it was used in crowdsourcing to compose tasks with different difficulty levels and test the impact on skill improvement and worker satisfaction [?]. *The difference with our work is that the composition of test difficulties is decided beforehand (for instance, by alternating easy and difficult tasks).*

Learner Modeling & Mastery Detection. Many works [?] develop criteria that determine if a learner mastered a skill. *NCC* (*N Consecutive Correct*) [?] declares mastery if the number of consecutive correct answers exceeds a threshold. *Moving Average* [?] declares mastery if the average of correct answers within a moving window exceeds a threshold. More sophisticated models were also proposed [? ?]. The two most popular are *Bayesian Knowledge Tracing (BKT)* [?] and *Latent Factor* models [? ? ?]. BKT [?] is a hidden Markov model with 4 parameters: probability that the skill is initially mastered, probability of learning in one iteration, probability of an incorrect answer when the skill is learned (slip), and probability of a correct answer when the skill is unlearned (guess). Many extensions were proposed [? ? ?]. For example, KT-IDEM [?] accounts for test difficulty. *Latent Factor* are based on logistic regression. They learn latent parameters to infer the probability of mastery using a sigmoid function.*We leverage KT-IDEM [?] to simulate learners. We also leverage NCC to update skills.*

6 CONCLUSION AND FUTURE WORK

We tackled the question of adaptive upskilling following a mastery learning approach. The originality of our approach lies in adapting the difficulty of tests to the learner's predicted performance, aptitude, and skill gap. However, it is worth mentioning that our framework has a number of limitations. First, it is tested on simulations. It would need to be deployed with user studies in a real-world system with real learners or on a crowdsourcing platform. Second, while the learner model (BKT) is largely equivalent to other models [? ?], our results might differ with the use of other models. Also, test difficulties are inferred and not defined by a domain expert. Finally, the condition of reaching skill mastery could be relaxed since in our case, learners might be able to reach high skill levels and still not be considered as mastering a skill.

For future work, we are investigating two research directions.

Reinforcement learning. One direction we are pursuing is the applicability of reinforcement learning to our setting. The idea would be to devise a reward function that reflects our optimization dimensions and learn a policy as a series of k tests. This will have the benefit of pre-training different policies and reducing waiting time at deployment. It will also provide the ability to incorporate more constraints into our framework in the spirit of our recent work [?]. Our aim is to explore the benefit of this global optimization and compare it to the step-wise optimization we proposed in this paper.

Collaborative learning in the physical world. There are many learning theories in the physical world, such as situated learning [?] and collaborative learning [?]. One representative of the former is apprenticeship where knowledge is propagated from experts to novice learners based on the principle of *Legitimate Peripheral Participation* [**?**]. Collaborative learning is also effective in online learning environments like MOOCs, and studies showed that rich interactions such as peer feedback and discussion promote learning [**? ? ?**]. Our framework could be extended to account for peer recommendation at each step.

REFERENCES