Multi-objective Test Recommendation for Adaptive Learning

Nassim Bouarour, Idir Benouaret, Sihem Amer-Yahia

Multi-Objective Test Recommendation for Adaptive Learning

Nassim Bouarour[1], Idir Benouaret[2], and Sihem Amer-Yahia[1]

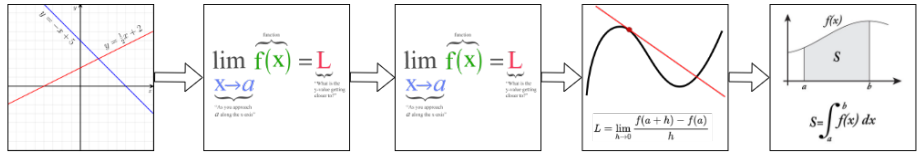
[1] CNRS, Univ. Grenoble Alpes, France
firstname.lastname@univ-grenoble-alpes.fr
[2] Epita, Lyon, France
firstname.lastname@epita.fr

Abstract. Upskilling is a fast-growing segment of the education economy [31]. Yet, there is little algorithmic work that focuses on crafting dedicated strategies to reach high-skill mastery. In this paper, we formalize AdUp, an iterative upskilling problem that combines mastery learning [49] and Zone of Proximal Development [7]. We extend our previous work [9] and design two solutions for AdUp: `MOO` and `MAB`. `MOO` is a multi-objective optimization approach that relies on Hill Climbing to adapt the difficulty of recommended tests to three objectives: learner's predicted performance, aptitude, and skill gap. `MAB` is a meta approach based on Multi-Armed Bandits to learn the best combination of objectives to optimize at each iteration. We show how these solutions are combined with two common learner simulation models: BKT (KT-IDEM) [47] and Item Response Theory (IRT) [53]. Our simulation experiments demonstrate the necessity of leveraging all three objectives and the need to adapt the optimization objectives to the learner's progression ability as `MAB` offers a higher mastery rate and a better final skill gain than `MOO`.

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 [45, 31]. Moreover, nowadays, 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 [22]. 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 [55] 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 [49] where the focus of instruction is the time required for different learners to acquire the same competencies and achieve the same level of mastery. To the best of our

knowledge, our work is the first to propose formalization that encounters mastery learning. This learning strategy is very much in contrast with classic models of teaching where all learners are given approximately the same time to learn. We illustrate that with an example.



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	

Fig. 1: Example of the process of learning mathematical functions.

Motivating example. Consider a learner with very basic knowledge in Math 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 she provides correct answers. As a result, she incurs no skill gap, and her skill is updated accordingly. This triggers a second step where she is assigned more difficult tests (on limits of functions) on which she fails. In addition to not increasing her skill, she incurs a skill gap. To fill that gap, she is given a second chance with the same type of tests on which she succeeds. Her input is correct and her skill is updated. The same process is repeated, and the learner receives more difficult tests on derivatives and then on integrals. She provides correct results and her 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 expected performance either by assuming tests with the same difficulty level [49] or by pre-defining the composition of difficulties beforehand (e.g., by alternating test difficulty levels [36]). 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 potential skill gap for determining the next k tests. This will motivate the learner to work on her weaknesses and previous failures. 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.

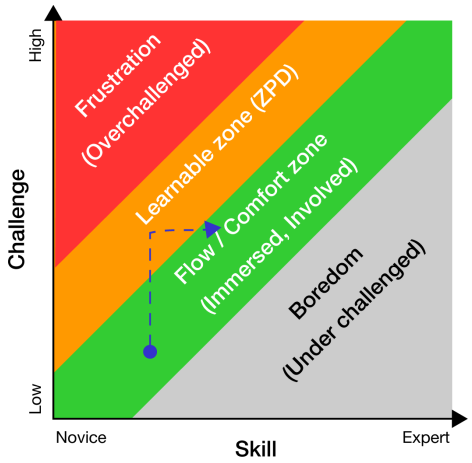


Fig. 2: Illustration of the combination of the Zone of Proximal Development (ZPD) and Flow Theory. In [7], it is shown that learners improve their skills by completing tests that are challenging but not too hard (dotted line).

Contributions. This paper represents an extension of our previous work [9]. We formalize AdUp (Adaptive Upskilling) as an optimization problem where a learner receives k tests that maximize expected performance and aptitude and minimize the accumulated skill gap. The combination of these objectives constitutes the novelty of our formalization.

The main challenge in solving AdUp, is its multi-objective nature. We propose to explore two solutions: a Multi-Objective Optimization, referred to as `MOO`, and a Multi-Armed Bandits solution, referred to as `MAB`. `MOO` is the natural formulation when all dimensions need to be optimized and is addressed by developing a Pareto solution that relies on dominance between k test sets and a *Hill Climbing* [42] heuristic algorithm that finds a subset of the non-dominated solutions [6]. Several variants can be drawn from `MOO` depending on the different compositions between the objectives. A drawback of `MOO` is that all variants optimize exactly the same dimensions over all the assigned batches of tests during the whole learning process. It would be desirable to have an approach that learns to find the dimensions to optimize at each iteration. For example, if the learner keeps providing wrong answers to the same tests, favoring the optimization of gap could be more desirable as we need to make sure that the learner successfully completes tests before providing more challenging ones. Therefore, we propose `MAB`, a solution that learns which of the three optimization dimensions to optimize at each iteration of k tests. We formalize this approach as a multi-armed bandit (`MAB`) problem.

Empirical validation. Our experiments examine the effectiveness of the optimization dimensions on upskilling. To investigate that, we divided our experiments into two parts. In the first one, we examine the impact of our solutions on mastery by simulating the learners' answers as well as the whole learning process. We formulate four research questions: **RQ1.** Is the combination of all optimization dimensions well-adapted for attaining mastery and improving skill gain? **RQ2.** Do different settings of the skill update strategy exhibit different results? **RQ3.** Does the choice of the learner simulation model impact mastery and skill gain? **RQ4.** Does an application of a meta-strategy that chooses to optimize a subset of dimensions at each iteration (`MAB`), improve mastery achievement? In the second part, we examine the quality of the assigned tests at the next iteration of the learning process based on the interactions of learners with real educational systems. We then formulate a research question **RQ5.** Does the optimization of all the dimensions lead to a more relevant test assignment compared to state-of-the-art baselines?

We use three real-world datasets: `MatMat` collected from a Czech educational system (matmat.cz)[3], `ASSISTment` challenge data collected from the ASSISTment platform[4], and, `ASSISTment2009` data that contains learners' answers to questions extracted from the same ASSISTment platform between 2009 and 2010. From each dataset, we infer the difficulty levels of the tests based on their features (e.g., type of the test) if they are available. If the features are missing, the inference is based on the correctness rate of the tests of all learners. To simulate learners and predict their probability of providing correct answers, we leverage two models: an extended version of *Bayesian Knowledge Tracing (BKT)* [14] that leverages test difficulties [47] and the *Item Response Theory model (IRT)* [53]. These models capture the learning process of learners and infer the tests that are correctly answered by them. After each iteration, the skill of a learner is updated following an existing approach (i.e., NCC [27]) that aggregates the learner's performance based on her N last consecutive answers. For example, when $N = 3$, the learner's skill is updated if she provides three consecutive correct answers on tests with the same difficulty level.

Summary of findings. For the first part of the experiments, we summarize the findings of each research question. On **RQ1**, we find that combining all objectives performs better than optimizing one or two dimensions only. We show that it yields the highest mastery in fewer iterations. Our results confirm the ZPD and Flow theories [7] and the importance of leveraging aptitude and challenging learners. Moreover, as our skill update strategy is based on NCC, we find, on **RQ2**, that `MOO` is not sensitive to the variation of the value of N. On **RQ3.**, we find that the results of **RQ1**, performed on BKT, are generalized when using another simulation model (i.e., IRT). We also observe that `MOO` offers the highest rate of mastery and optimizing aptitude remains essential. The main difference between the two simulation models is that IRT tends to favor the minimization of gap while BKT favors the maximization of expected performance. On **RQ4.**,

[3] https://github.com/adaptive-learning/matmat-web/blob/master/data/data_description.md
[4] https://sites.google.com/view/assistmentsdatamining

we find that choosing automatically the combination of dimensions to optimize at each iteration improves both the skill gain and the percentage of learners that attain mastery. This confirms the need for a meta-strategy that automatically adapts to the learner's skill progression. For the second part of the experiments, we find, on **RQ5.**, that `MOO` is not only the best in terms of upskilling but it offers the possibility to a larger number of learners to improve their skill compared to recommendation-based (e.g., KNN [56]) and knowledge-based baselines [51, 66](i.e., models that capture the learning and the knowledge of learners). This shows that our optimization dimensions capture with accuracy the evolution of the learning process of a given learner. It also shows that our solution can adapt to different learners.

Organization. In Section 2, we define our data model as well as the optimization dimensions we study and we give a formalization of the AdUp problem. Section 3 provides a description of our algorithms. Our extensive experiments are described in Section 4. We provide a review of the related work in Section 5. We conclude with a summary and discussion on future work in Section 6.

2 Model and Problem

We consider a learner $l \in \mathcal{L}$ who follows an iterative learning process for a skill sk. We focus on one skill that has a scalar value. Extending the skill representation to a vector is not straightforward. It requires studying independence between skills or making an independence assumption which may be unrealistic.

At each step, l completes a set of k tests with different difficulty levels for sk. Each test $t \in \mathcal{T}$ has a fixed difficulty d_t. 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 consider that a learner attains mastery when hey skill value $l.sk$ can not be further improved and is equal to the highest difficulty level.

We aim to formalize a problem where at any given iteration, the learner receives a batch of k tests whose difficulty level is strictly 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 skill sk when assigned tests that are correctly completed. Aptitude is defined as follows:

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

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.:

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

3 Solutions

The main challenge in solving AdUp is its multi-objective nature. Scalarization is a common approach that transforms the problem into a single objective whereas optimization dimensions are combined via a weighted linear sum. Another approach is the ϵ-Constraint method where a single objective is optimized and the other objectives are constrained with user-specific values [46]. These methods suffer from the need to fix weights or thresholds, leading to sub-optimal solutions. Therefore, we propose to explore two solutions: a Multi-Objective Optimization, referred to as `MOO`, and a Multi-Armed Bandits, referred to as `MAB`.

3.1 Multi-Objective Optimization (`MOO`)

We propose an approach that finds the Pareto solutions by addressing all objectives at once [6]. To do so, we define a dominance relation between two sets of tests of size k.

We represent the set of all test batches as $C_k = \{B | B \subseteq \mathcal{T}, |B| = k\}$. 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, based on [42], to avoid an exhaustive exploration of the search space of possible solutions. It starts by performing *times* iterations where in each it finds an optimal batch of tests (Lines 3 to 7) to avoid local optima. At each iteration, it first generates a random set of k tests. Then it performs *Hill Climbing* to optimize both expected performance and aptitude using Algorithm 2. The Hill Climbing optimizes these two dimensions because they may be related. Indeed, the tests with a high aptitude tend to have lower expected performance (See Figure 2). 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 gap is chosen (Line 9) and assigned to the learner (Line 10). The skill gap is not directly optimized within Hill Climbing as it is independent of both aptitude and expected performance. The learner's skill is updated after the completion of the test batch (Line 11). Refer to Section 4.2 for our skill update strategy. This process is repeated until the learner l masters the skill (i.e., correctly answering the most difficult test).

Algorithm 1: Heuristic MOO that optimizes Aptitude, Expected Performance, and Gap

```
   Input: learner l, set of tests 𝒯, size k, # repetition times
 1 while not mastery do
 2 |   Results ← ∅
 3 |   for n in [1..times] do
 4 |   |   C ← Random_candidate(k)
 5 |   |   C* ← HCAE(C) \\Algorithm 2
 6 |   |   Results.Add(C*)
 7 |   end
 8 |   Keep non-dominated candidates in Results
 9 |   B ← The solution from Results with the lowest skill gap
10 |   l completes B
11 |   l.sk ← skill_update(l.sk, B)
12 end
```

Algorithm 2 is a routine that is called from Algorithm 1 and 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 next 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.

`MOO` variants. There are multiple solution variants to AdUp: `MOO` as described in Algorithm 1; `MOEG`, `MOAG`, and `MOAE` optimize expected performance and gap,

Algorithm 2: HCAE - Hill Climbing for Aptitude and Expected Performance (Called from Algorithm 1)

Input: Batch of k tests B
Output: Optimized batch B^*

```
1  while True do
2      Candidates ← ∅
3      for test ∈ B do
4          test_down ← A test with the next lower difficulty
5          B_1 ← B − {test} + {test_down}
6          test_up ← A test with the next higher difficulty
7          B_2 ← 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 ← A random candidate from Candidates
17     end
18 end
```

aptitude and gap, or aptitude and expected performance respectively; `MOG`, `MOE`, and `MOA` optimize gap only, expected performance only, or aptitude only respectively. Similarly to Algorithm 1, the bi-objective variants are also based on *Hill Climbing* to explore the space of batches and find potential candidates. In the case of `MOAE` and `MOEG`, the *Hill Climbing* optimizes expected performance. The condition in Line 9 (Algorithm 1) relates to aptitude for `MOAE` and gap for `MOEG`. On the other hand, for `MOAG`, the *Hill Climbing* optimizes aptitude, and Line 9 remains unchanged.

3.2 Multi-Armed Bandits Algorithm (`MAB`)

A drawback of the previous solution is that all the variants optimize exactly the same dimensions over all the assigned batches of tests during the whole learning process. However, it would be desirable to have an approach that can learn to find the dimensions to optimize at each iteration. For example, if the learner keeps providing wrong answers to the same tests, optimizing gap solely could be more desirable as we need to make sure that the learner successfully completes these tests before providing more challenging ones. On the contrary, if the learner answers correctly the last batches of tests, it might be better to optimize aptitude so that the learner gets challenged with more difficult tests as she has no gap in her learning process. Therefore, our goal is to design an approach that chooses

automatically which of the three dimensions will be optimized at each iteration of k tests. We formalize this approach as a multi-armed bandit (MAB) problem.

The goal of MAB is to verify if a meta approach could be used to address the AdUp problem. The meta approach chooses, at each iteration, an optimization variant of our problem, i.e., bi-objective, or multi-objective optimizations, to generate k tests. We formalize that as a multi-armed bandit problem where each arm corresponds to an optimization variant and the reward r_i, at iteration i, for each variant v is defined as the speed of skill progression:

$$r_{iv} = \frac{\sum_{\forall \text{iterations } j, j<i} \text{skill gain offered by } v \text{ at iteration } j}{\#\text{time the variant } v \text{ was chosen}}$$

At each iteration, the skill progression speed of each arm is computed and the one with the highest cumulated progression is selected. In the case where an arm has never been selected before, its speed is set to zero. The batch of k tests is then generated based on the variant of the chosen arm.

MAB variants. We implemented different multi-armed bandit strategies [58]: ϵ-GREEDY that chooses randomly an arm (i.e., variant) with an ϵ probability. It chooses the arm with the highest reward with a 1-ϵ probability. THOMPSON Sampling which selects the arm with the highest probability that is learned from previous interactions. The third strategy is the upper confidence bound (UCB) which combines the reward and an uncertainty measure with a confidence degree (c) that balances between exploitation and exploration. Finally, the SOFTMAX strategy relies on Boltzmann distribution that has a parameter (τ) that specifies the randomness of the exploration to choose the optimal arm.

4 Experiments

In this section, we conduct extensive experiments to show the effectiveness of our proposed solutions. We divide our experiments into two parts. In the first part, we compare variants of both MOO and MAB to select the best combination of optimized dimensions to provide the best learning experience to users. We evaluate the overall sequence of assigned test batches from the beginning of the learning process until attaining mastery by simulating the learners' answers. In the second part, we evaluate the quality of the generated batches, at one iteration of the process, based on interactions of real learners. We compare our variants to state-of-the-art adaptive learning and recommendation models. In the following, we first introduce the datasets that we use. We then describe our skill update strategy and finally present the settings and results of each part of the experiments.

4.1 Datasets

We use three real-world datasets that we summarize in the following.

- `MatMat`[5]: The data is collected from a Czech educational system. 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 consider 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. Within each difficulty range, we assume that multi-digit operations are more difficult than single-digit ones, and tests displayed with visualisations are simpler than directly written tests.

- `ASSISTment` Challenge [6]: The data contains information about the free online tutoring ASSISTment platform [7]. The dataset is composed of school math exercises sampled from the Massachusetts Comprehensive Assessment System (MCAS) containing different types of tests. From the different versions of datasets, we used the ASSISTment challenge one. It contains more than 3000 tests answered by 1709 students. From the data, we selected 10 types of tests (e.g., additions, fractions) and inferred, using the same procedure as `MatMat` and based on the features of the tests, 26 distinct difficulties.

- `ASSISTment2009`[8]: The data also contains information about the free online tutoring ASSISTment from the 2009-2010 year. We use a subset of the data that was extracted by [35]. The data contains more than 17700 tests answered by more than 4000 students. We classified the tests into different classes and each class has a difficulty level. As the data does not contain tests' features, they are classified by their overall rate of correct answers. We consider the tests with a high rate of correctness the easiest.

4.2 Skill update and mastery achievement

At each iteration and after the completion of a batch B of k tests, we update the skill of learner l as follows:

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

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 (Table 1). Using our

[5] https://github.com/adaptive-learning/matmat-web/blob/master/data/data_description.md
[6] https://sites.google.com/view/assistmentsdatamining
[7] https://new.assistments.org
[8] https://sites.google.com/site/assistmentsdata/home/2009-2010-assistment-data/skill-builder-data-2009-2010?authuser=0

Test	Difficulty	Learner Input
t_4	0.35	True
t_5	0.4	False
t_6	0.45	True

Table 1: Test completion examples.

strategy, the skill value $l.sk$ is updated to 0.35 (difficulty of t_4). The correct completion of t_6 is not considered as there exists one test (t_5) with a lower difficulty that was wrongly completed. To account for variability in learners' answers, we used the static mastery detection method NCC [27] that updates the skill if the number of consecutive correct answers for a given difficulty level is N. For mastery achievement, we consider that learners attain mastery when their skill can not be further improved and is equal to the highest difficulty level.

4.3 Research Questions

Our goal is to address the following research questions related to the performances of `MOO` and `MAB`:

- **RQ1:** Is the combination of all optimization dimensions well-adapted for attaining mastery and improving skill gain?
- **RQ2:** Do different settings of the skill update strategy exhibit different results?
- **RQ3:** Does the choice of the model of learner simulation impact mastery and skill gain?
- **RQ4:** Does an application of a meta-strategy that chooses to optimize a subset of dimensions at each iteration, improve mastery achievement?
- **RQ5:** Does the optimization of all the dimensions lead to a more relevant test assignment compared to state-of-the-art baselines?

We divide these questions into two parts. The first part contains the four first questions where we evaluate the whole learning process of learners (i.e., from the beginning of the process until attaining mastery). To do so, we simulate the learners and their answers. The second part contains only the last question where we evaluate the quality of the assigned test batches on real learners instead of simulating them.

In the following, we first present the models used for simulating the learners. We then present the different metrics, baselines, and experimental settings. We finally report the results to answer each RQ.

4.4 Experimental Settings

Learner simulation. To answer the first four questions, we need to simulate the answers of the learners. From all the existing simulation models, we rely on

these established ones:

• ***Bayesian Knowledge Tracing (BKT).*** This model simulates learners using an extended version of BKT (KT-IDEM) [47] that takes into account the difficulty level of tests. BKT is a cognitively diagnostic form of assessment that has been recognized as beneficial to learners and instructors [47]. It 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, two more probabilities are used to estimate the performance of the learner: Guess and Slip. 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. We use the implementation of [5] in our experiments.

• ***Item Response Theory (IRT).*** This model simulates learners based on latent factors [11]. The probabilities of the next tests are calculated by applying a sigmoid function and learning a logistic regression to predict responses of learners. One method, AFM [11], infers the probability by characterizing the learner and the difficulty of tests with two distinct parameters. Another method, PFM [48], extends AFM by integrating the number of successes and failures as parameters in addition to previous ones. Other latent models are based on Item Response Theory (IRT) [53], a traditional cognitive diagnosis model [30]. The simplest version [53] predicts a probability of a binary answer (correct/incorrect) by assuming a unique internal parameter for each learner. In addition, it defines tests with one parameter (difficulty) [52], two parameters (the number of correct answers and difficulty) [8], or three parameters (probability of correct answer) [34]. In our experiments, we used this last method based on the implementation of [59]. The reason is that compared to other latent models, it incorporates the probability of guessing in addition to the difficulty and the number of correct answers.

BKT and IRT are structurally different as BKT captures the learning as a chronological process while latent models do not capture the temporal dimensions. They are trained differently as BKT uses the Expectation Maximization (EM) algorithm [5] and IRT uses Adam [59]. Despite these differences, both BKT and latent models infer the probability of correct answers and simulate the learning by capturing similar concepts: the difficulty of tests, the level of learning, and the probability of guessing the correct answers.

Variants. In our experiments we used two types of variants. In the simulation part, we compare `MOO` and its variants described in Section 3.1 as well as `MAB` and its variants described in Section 3.2. We recapitulate them in Table 2. In the real-learners part (i.e., **RQ5**), our goal is to verify whether our dimensions, introduced in Section 2.1, capture well the knowledge of the learners. We then

use the MOO solution that combines all three of them. We define two variants of the solution: the original MOO, presented in Section 3.1, that assigns batches that only contain tests with higher difficulties and MOO_BEG that assigns batches of tests for which the difficulties are lower than the skill level of the learner.

Variant		Description
MOO	MOO	optimize expected performance, aptitude and gap
	MOAE	optimize expected performance and aptitude
	MOAG	optimize aptitude and gap
	MOEG	optimize expected performance and gap
	MOE	optimize expected performance only
	MOA	optimize aptitude only
	MOG	optimize gap only
MAB	ϵ-GREEDY	choose randomly an arm with an ϵ probability
	SOFTMAX	rely on Boltzmann distribution
	THOMPSON	sample the arm with the highest probability
	UCB	combine the reward and uncertainty with a confidence degree (c)

Table 2: Recapitulation of the variants of our solutions

Baselines. As we want to simulate the whole learning process, we consider ALTERNATE, a state-of-the-art approach that assigns a random set of k tests whose difficulty levels alternate in a round-robin fashion: k easy then k medium then k hard tests [36].

Moreover, to verify the quality of our solution on real learners, we compare it to state-of-the-art methods that capture the knowledge of learners and its evolution. Different models [2] were proposed in the literature. These models differ in their assumptions about the way they represent and quantify the learners' knowledge. They also differ in their strategy of tracking the learning progression. As they can capture the knowledge state of the learners, they were also used in the literature for test assignments and recommendations [26]. In this work, we consider a wide range of methods covering both recent and mostly used ones. We give a brief description for each one:

- KNN [56]: k nearest neighbors. This baseline is commonly used for recommendations. As adaptive upskilling can be seen as a recommendation problem, we used this method as a baseline to verify how standard recommenders behave for upskilling. KNN finds a predefined number of tests that are similar to the learning path of the learner using cosine similarity. The k tests that have a high similarity are the assigned ones.

- BKT [47]: Bayesian Knowledge Tracing. As explained in more detail in Section 4.4, this baseline models the knowledge of learners given the chronological learning path of the learner. It computes the probability of correctly

answering a test. For a given learner, the batch of tests that maximizes these probabilities is assigned.

- `IRT` [53]: Item Response Theory. We also introduced this baseline in Section 4.4. It is a popular solution that estimates the performance of learners by learning a logistic function. For a given learner, the batch of tests that maximizes the performance is assigned.

- `MCD` [59]: Matrix-Factorization Cognitive Based. This method applies the matrix factorization method to education-related data. It embeds both learners and tests and represents the link between them in a latent space. The model learns the latent space based on the learning path of the learner.

- `NCDM` [63]: Neural Cognitive Diagnosis Model. It is a recent model that incorporates neural networks to learn the learner-test interactions. It projects learners and tests to factor vectors and captures knowledge relevancy and proficiency. It leverages multi-neural layers to output a predicted score of the correctness of a test by a given learner.

- `DKT` [51]: Deep Knowledge Tracing. It is an extension of the original BKT. It relies on Recurrent Neural Networks (RNNs) [32] to model the learners' process and predict their probability of correctly answering a test. The tests are ranked based on their probability.

- `DKVMN` [66]: Dynamic Key-Value Memory Network. It uses a key-value memory network [37] to capture the learners' knowledge state and its evolution. It is composed of two matrices: the key that stores the representation of test difficulties and the value that stores the knowledge level of the learner. In this model, the key memory is static while the value one is dynamic (updated after each iteration). The model produces the probability of correctly answering a test.

- `SAKT` [44]: Self-Attentive Knowledge Tracing. This model adds an attention mechanism to the original knowledge tracing models. It uses the mechanism proposed by [61] to learn attention matrices. It also incorporates multiple attention heads. Each attention matrix learns the importance of a test in the past interactions of the learned in predicting the correctness of the current test. It also predicts the probability of answering correctly a test.

We can group the selected baselines into three categories: Recommendation-based baselines (`KNN`), Traditional knowledge tracing (`BKT`, `IRT`, `MCD`), and Advanced knowledge tracing (`NCDM`, `DKT`, `DKVMN`, and `SAKT`). In the last category, we selected the models based on their internal structure of capturing the knowledge and its evolution (Deep Learning, Recurrent Learning, Key-Value Network, and Attention Mechanism).

Metrics. We report (1) the average skill gain i.e. the difference between the last and first skill values for all simulated learners, and (2) the average skill progression i.e. the average skill evolution from iteration to iteration. 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.

We want to report the quality of the generated test batches for real learners. For this reason, we evaluate both the logic [67] of the assigned test batches wrt the previous interactions of the learner and the potential knowledge gained when correctly performing these tests. To capture the logic of the assigned batch, we rely on relevance metrics. These metrics are learner-based and evaluate the proportion of the tests that were assigned and were truly performed by the learner in real world (i.e., relevant tests). We then report (6) Precision i.e. the proportion of assigned tests that are relevant, (7) Recall i.e. the proportion of relevant tests that are assigned, and (8) F-Score i.e. a combination between Precision and Recall that captures the balance between them. Finally, we report (9) the percentage of learners who improved their skills under each variant or baseline.

Parameters. In the first four questions, we evaluate the overall sequence of batches assigned to the learners from the beginning until attaining mastery. We set the maximum number of iterations to attain mastery to 500. We vary the value of k in $\{3, 5, 10, 15, 20\}$ and the number of simulations, i.e., learners, in $\{50, 100\}$. We only report results of 100 simulations. Results on other settings are similar.

On the other hand, in **RQ5**, we evaluate the quality of assigned batches for real learners at one iteration of the process. This iteration is chosen based on the split of the data. In fact, as selected baselines need training, we split our data in a learner-wise fashion, i.e., the split is done for every learner, chronologically according to the provided timestamps so that 70% of the data constitutes the training set and the remaining 30% is assigned to the test set. In the case where the timestamps are not provided, the split is random.

We rely on this GitHub repository [9] for the implementation of `NCDM`, `DKT`, `DKVMN`, and `SAKT`. We used the same training configurations and the same values of the hyperparameters of the models. We also rely on [59] for the implementation of `IRT` and `MCD`). We refer the reader to our GitHub repository[10] for our complete results and code for reproducibility.

4.5 RQ1: Impact of optimizing all dimensions

To verify the impact of optimizing all dimensions, we use BKT (KT-IDEM) [47] and assume $N = 1$ in the skill update strategy. We consider two datasets: `MatMat`

[9] https://github.com/hcnoh/knowledge-tracing-collection-pytorch/tree/main
[10] https://github.com/AdaptiveUpskilling/AdUp.git

and ASSISTment2009 challenge. We consider two settings: fixed initial skill value and variable initial skill value.

• **Fixed initial skill value.** We assume the same fixed initial skill value for all learners and consider that learners attain mastery when their skill equals the highest difficulty level. We set the initial value to the lowest difficulty level in our simulated data.

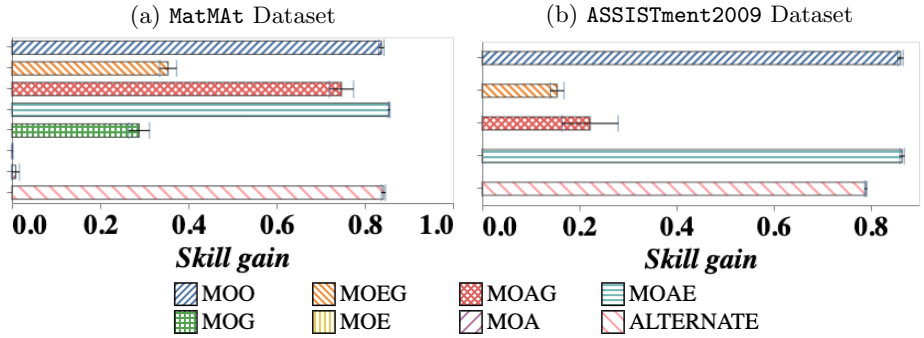


Fig. 3: Average skill gain for each variant with a fixed initial skill.

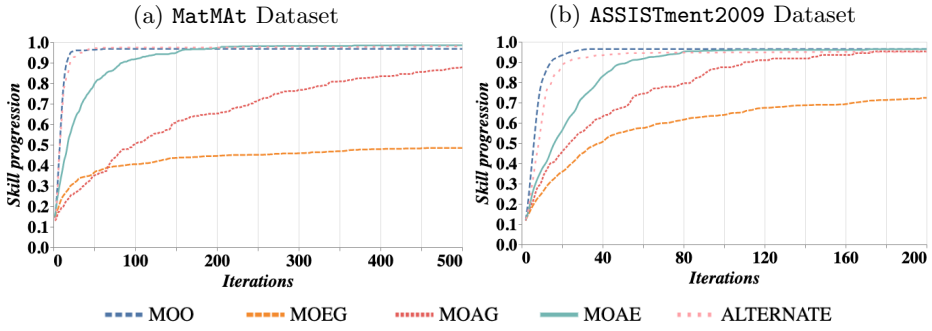


Fig. 4: Skill progression as a function of # iterations with a fixed initial skill.

Skill gain and progression. Figure 3 reports the average skill gain. We observe that MOO and MOAE produce the highest average skill gain for both MatMat and ASSISTment2009 datasets. Surprisingly, ALTERNATE seems to also produce a high skill gain outperfoming in both cases MOAG and MOEG. 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 but still faster than MOEG. This reinforces our initial assumption that optimizing for all three objectives at once yields the best results. It also shows that alternating task difficulties does yield good skill gain and progression. Therefore, in the next experiment, we examine whether ALTERNATE compares favorably to MOO and MOAE in terms of achieving skill mastery.

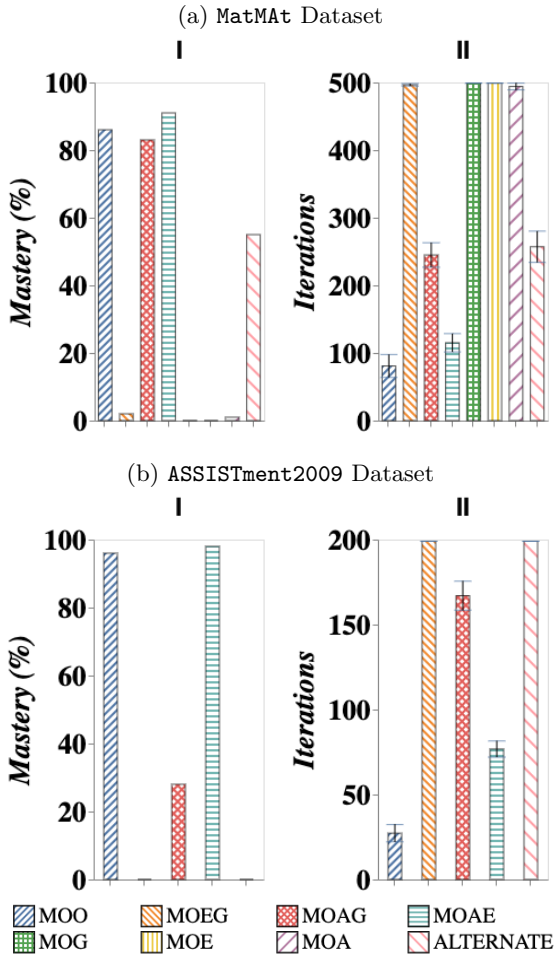


Fig. 5: **(I)** Percentage of learners who attain mastery - **(II)** Average number of iterations to attain mastery.

Mastery. Figure 5 (I) reports the number of times each variant attained mastery. One can see that while `ALTERNATE` reaches a reasonable mastery level in `MatMat` ($\approx$ 59%), it is much lower than `MOO`, `MOAG` and `MOAE` ($\approx$ 90%). On the other hand, one can see that on `ASSISTment2009`, `MOO` and `MOAE` remain effective while `ALTERNATE` mastery rate is null. This clearly confirms that aptitude plays a central role in attaining mastery as all variants that optimize it offer higher mastery rates than `ALTERNATE`. Hence, while alternating test difficulty levels do achieve good skill gain (Figure 3) and skill progression (Figure 4) 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 [62]. `MOA` assigns tests that are too difficult and that keep the learner in a frustration zone [62].

Figure 5 (II) 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` in `MatMat` 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 that 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. Finally, the figure shows that `MOAE` attains a slightly higher mastery level than `MOO` in both datasets, but it is clearly outperformed by `MOO` in terms of the number of iterations needed to achieve that mastery level.

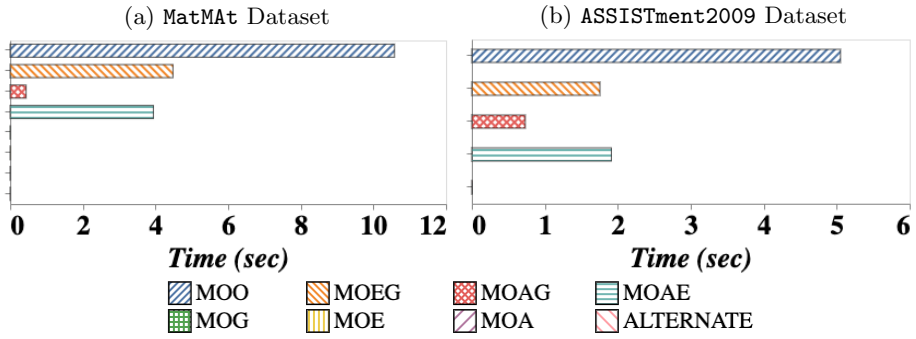


Fig. 6: Average time for generating one batch.

Response time. Time experiments have shown that single-objective variants are obviously the fastest to generate a batch of k tests (Figure 6). `MOO` has the worst time average as it has to optimize three objectives ($\approx$10 seconds for `MatMat` and $\approx$5 seconds for `ASSISTment2009`). `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.

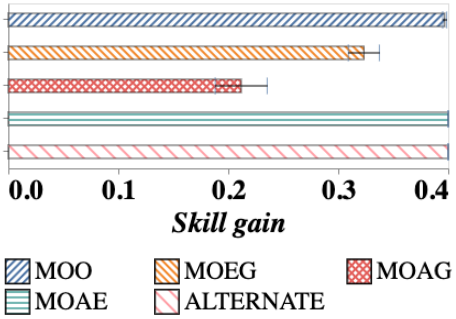


Fig. 7: Average skill gain with variable initial skills on MatMat.

• **Variable initial skill value.** We study the case where learners have different initial skill values and consider that a skill is mastered when the skill gain attains a fixed value. We set the value to 0.4 as it is the highest skill gain in common between all learners. We only present results on MatMat as similar observations are made on ASSISTment2009. We report only bi-objective and MOO variants in addition to ALTERNATE as we showed already that single-objective solutions are inefficient.

Skill gain and progression. From figure 7, which reports the average skill gain for all variants, we note that similarly to the case of a fixed initial skill value, MOO, MOAE, and ALTERNATE offer the highest skill gain that is equal to the maximum value (0.4). Figure 8 also generalizes previous results by showing that MOO and ALTERNATE skill progressions are the fastest followed by MOAE.

Mastery. Figure 9 shows the percentage of mastery attained by each variant as well as the number of iterations needed to attain mastery. One can confirm that, despite a smaller gain value to attain mastery, optimizing aptitude is still necessary as MOEG is the worst performer for the number of iterations and the second worst for mastery. We also see that ALTERNATE has comparable results to MOO and MOAE which confirms that it is capped in terms of mastery. Obviously, we can see that all variants attain mastery more often and in fewer iterations than when initial skills are fixed. This is due to the fact that in the latter, learners must achieve a much higher skill gain to attain mastery.

Findings. This experiment shows that combining all objectives yields the highest skill gain which permits a higher mastery in fewer iterations independently of the initial skill value of the learners. It also shows that challenging learners and

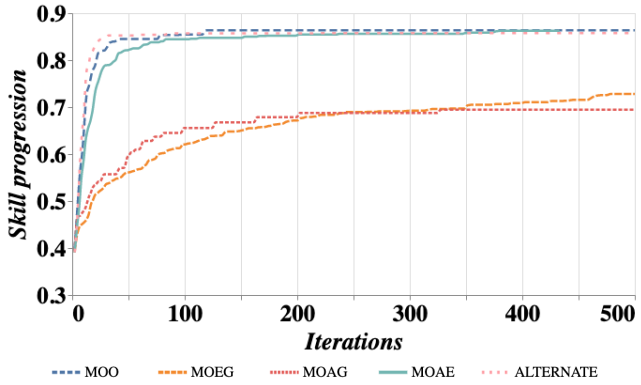


Fig. 8: Skill progression as a function of # iterations with variable initial skills on `MatMat`.

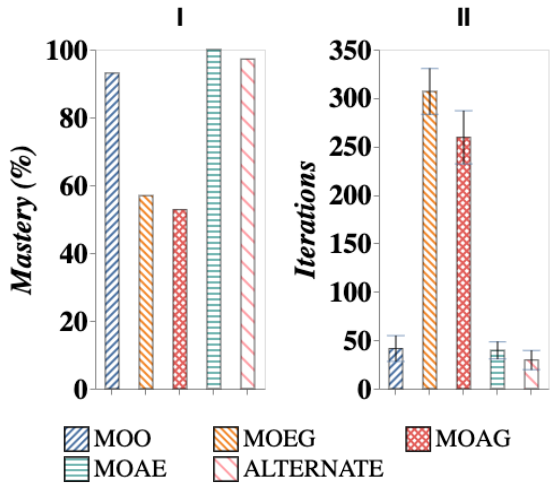


Fig. 9: **(I)** Percentage of learners who attain mastery - **(II)** Average number of iterations to attain mastery using variable initial skills on `MatMat`.

optimizing aptitude is beneficial to attain mastery. These results were observed on two different datasets `MatMat` and `ASSISTment2009` that have different characteristics. These results also confirm the ZPD and Flow theories [7] and show the importance of leveraging aptitude and challenging learners.

4.6 RQ2: Impact of changing the settings of the skill update

We report skill and mastery results by further challenging the learners during the skill update. We increase the value of N, the number of consecutive correct

answers, to $N = 3$. We report results in the case where the initial skill is fixed for both `MatMat` and `ASSISTment2009`.

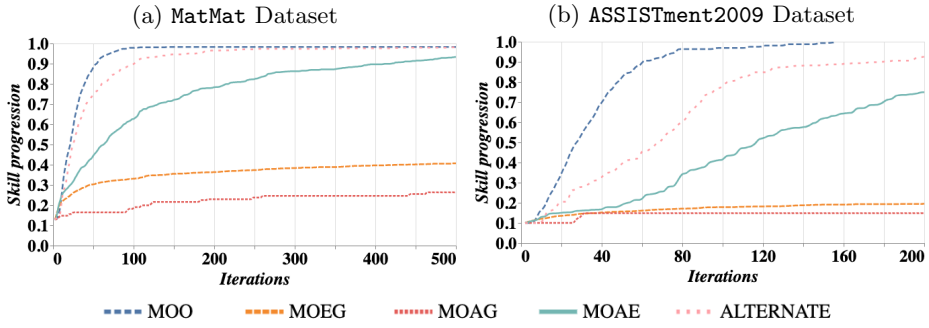


Fig. 10: Skill progression as a function of # iterations with $N = 3$.

Skill gain and progression. The average skill gain is similar to the one reported in Figure 3 where `ALTERNATE` is comparable to `MOO` and `MOAE`. `MOO` is the best variant while `MOAE` is the second best. Figure 10 shows the average progression of the skill on both datasets. We see that the progression is slower than the one presented in Figure 4 because the skill gain is slow. This is intuitive as learners have to answer correctly $N = 3$ tests of the same difficulty level to see their skill updated while previously one correct answer was enough. The second observation is that `MOO` is still the best variant with a clear advantage compared to `ALTERNATE` and `MOAE`. This means that `MOO` is less affected by the different values of N than other variants. To be sure of this conclusion, we study the Mastery rate and number of iterations under the $N = 3$ constraint.

Mastery. The results, in Figure 11, show that more than 90% of learners attain mastery under `MOO` while less than 70% achieve it under `ALTERNATE` and `MOAE` on `MatMat`. On the other hand, 99% of learners attain mastery under `MOO` on `ASSISTment2009`. We also see a small decrease in the mastery rates of `MOAG`, `MOEG` and `ALTERNATE`. Results also show that `MOO` is the fastest as it offers learners fewer iterations to reach the highest difficulty level on both datasets. These results confirm that `MOO` is not affected by different settings of the skill update strategy and is the best variant.

Findings. This experiment finds that `MOO` is not sensitive to varying different settings of the skill update strategy and that holds for all datasets.

4.7 RQ3: Impact of changing the learner simulation model

We report the results of the same metrics using a different learner simulation. We used item response theory (IRT) as explained in Section 4.4. We report results,

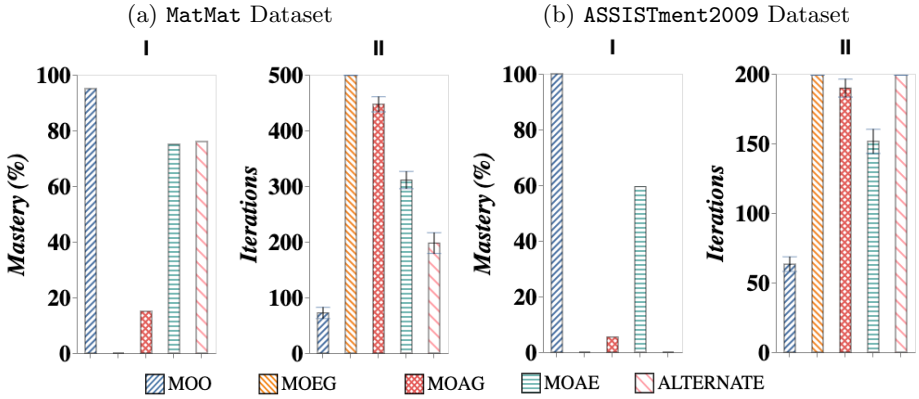


Fig. 11: **(I)** Percentage of learners who attain mastery - **(II)** Average number of iterations to attain mastery with $N = 3$.

on MatMat only, where the initial skill value is similar for all learners. Similar results were observed when the initial skill value was different from one learner to the other.

Skill gain and progression. Figure 12 reports the average skill gain for the variants that performed well previously. We observe that MOO and MOAG along with ALTERNATE produce the highest skill gain. One can also note that, similarly to the case of KT-IDEM, MOEG is the worst bi-objective variant. The main reason is that the test batches of MOEG do not challenge the learners as MOEG does not optimize for aptitude.

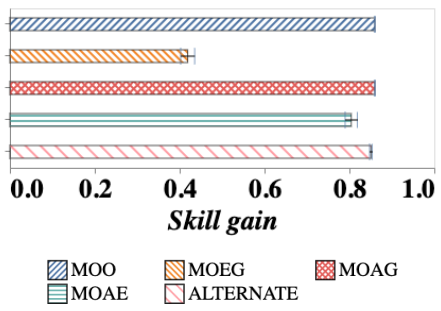


Fig. 12: Average skill gain using IRT on MatMat.

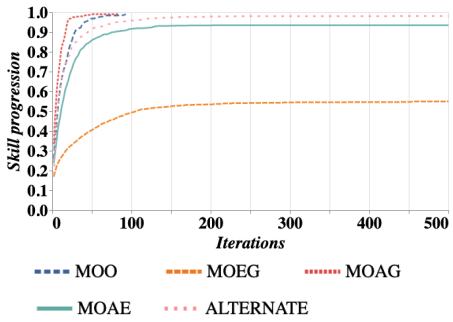


Fig. 13: Skill progression using IRT on MatMat.

Figure 13 shows the step-wise skill progression. We observe that MOO and MOAG are the fastest in terms of upskilling outperforming ALTERNATE which was

equivalent to MOO under BKT. One can also observe that MOEG is the slowest. Next, we compare these variants in terms of mastery.

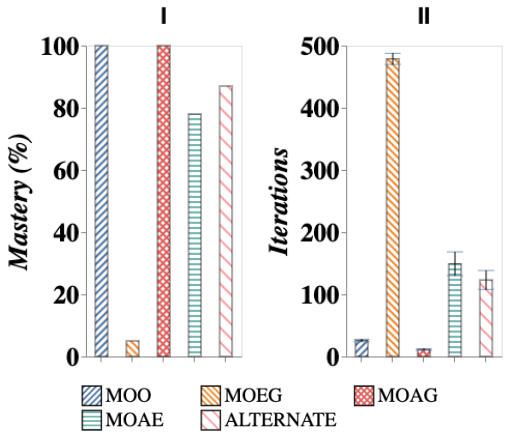


Fig. 14: **(I)** Percentage of learners who attain mastery - **(II)** Average number of iterations to attain mastery using IRT on MatMat.

Mastery. Figure 14 shows the average rate of mastery achieved by each variant as well as the average number of iterations to attain it. From the figure, we observe that the state-of-the-art alternating solution (ALTERNATE) achieves a high mastery level (≈80%) but it is clearly outperformed by MOO and MOAG. This experiment confirms that aptitude is required to attain a high rate of mastery as we see that MOEG is the worst variant. It attains mastery in ≈7% of time. From this figure, we can also observe that MOAE is outperformed by MOAG while it was better under the BKT model. A hypothetical explanation is related to the internal design of both methods. BKT formalizes the learning process as a hidden Markov model where test completion is viewed as a chronological sequence and where the different parameters are learned using the correctness of tests. In this case and intuitively, learner performances on recently assigned tests appear to be more influential than older tests while in the case of IRT, and because of the absence of time dimension, all performances have the same weight. Usually, as the gap is related to earlier tests, IRT seems to give more attention to it than BKT. Another possible explanation is that BKT tends to overestimate the importance of failure as reported in [48]. In that work, it was observed that BKT tends to predict worse performance after an incorrect answer. Based on that, one can make a hypothesis that BKT is negatively biased towards gap in contrast to latent factors models.

Results from Figure 14 (II) are inversely proportional to the ones depicted in Figure 14 (I). Variants with the highest mastery percentage are the quickest

to attain it. Inversely, the variants that attain a lower rate of mastery are the slowest. This indicates with more evidence that `MOO` is the best variant.

Findings. This experiment finds that IRT generalizes the results of KT-IDEM. In this case, we also observe that `MOO` offers the highest rate of mastery. Optimizing aptitude remains essential as `MOEG` is the worst variant. Despite the differences between BKT and IRT, one can explain their similar results with the fact that they both assume a guessing probability of correct answers and characterize tests by their difficulties. They both infer the correctness probability by approximating the knowledge of the learner based on previous correct answers (See section 4.4). In addition, prior work [23, 54] has shown that these models exhibit similar prediction accuracy. However, from these results, we see that the main difference between the two learner simulation models is that IRT tends to favor gap as `MOAG` is comparable to `MOO` while BKT favors expected performance as `MOAE` was the second best. We believe that further research needs to perform a more detailed comparison to understand why BKT and IRT offer the same predictions.

4.8 RQ4: Impact of the meta-strategy

We seek to verify whether choosing automatically a subset of learning dimensions to optimize at each iteration improves mastery and skill progression compared to optimizing fixed dimensions throughout the process. We implemented the four MAB strategies described in Section 3.2 and tested them with $N = 1$ as we showed in **RQ2.** that the value of N has no impact on the performances of our solution. We also assume a fixed initial skill value as we showed in **RQ1.** that variable initial skill exhibits similar performances. We used `MatMat` and `ASSISTment2009` for this experiment.

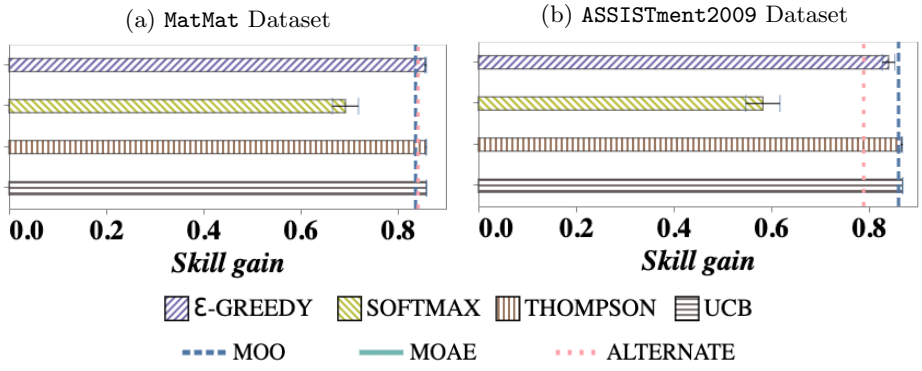


Fig. 15: Average skill gain using `MAB` strategies.

Skill gain and progression. Figure 15 shows the skill gain offered by the different MAB strategies. The lines represent the skill gain attained by learners under MOO and ALTERNATE. One can see that UCB and THOMPSON strategies slightly improve skill gain compared to MOO and ALTERNATE. We also see that SOFTMAX is the worst strategy showing that probability-based MAB is not adapted to this context in both datasets. Similar results can be observed in terms of skill progression in Figure 16. We can see that UCB is as fast as MOO. We can also see that THOMPSON has a better progression than ALTERNATE and MOAE, especially after a few iterations. Finally, one may note that ϵ-GREEDY has a similar progression than MOAE on the MatMat and a slightly slower on the ASSISTment2009. A possible explanation of these results is that ϵ-GREEDY takes a longer time to converge than the other MAB variants as it explores more especially at the beginning of the process. Moreover, we explain the outperformance of UCB by the fact that this variant converges quickly and always finds the right dimensions to leverage at each iteration.

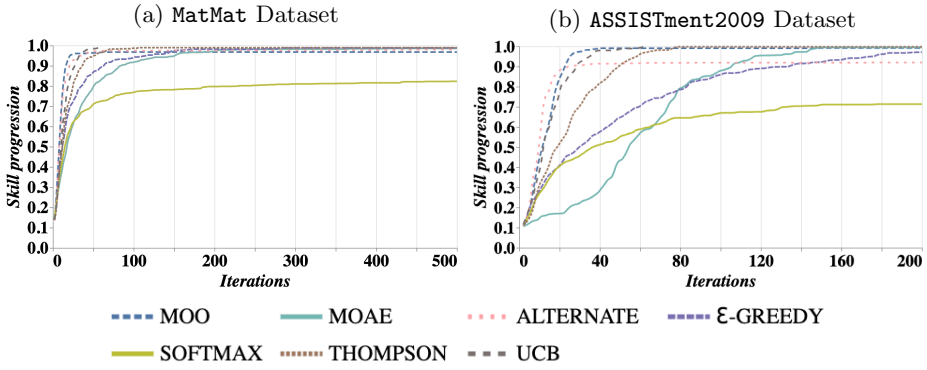


Fig. 16: Skill progression of learners using MAB strategies.

Mastery. Figure 17 shows the percentage of learners that attained mastery and the average number of iterations to achieve that. The lines represent the results of MOO, MOAE, and ALTERNATE. One can see that UCB is the best performer and outperforms all other MAB strategies as well as previous variants for mastery on both MatMat and ASSISTment2009. It also achieves that in fewer iterations. We can also see that THOMPSON attains more mastery than ALTERNATE, MOAE, and MOO in both datasets. It is also better than the two first ones but is equivalent to MOO in terms number of iterations in ASSISTment2009. In addition, ϵ-GREEDY variant is slightly outperformed by MOO in ASSISTment2009 data but is better in terms of mastery and iterations on the second one (MatMat). Finally, even if SOFTMAX is outperformed in terms of skill gain and progression, it achieves higher mastery and in a lower number of iterations than ALTERNATE in both datasets.

These findings add more evidence to the skill gain and progression results and confirm our previous assumption that selecting the dimensions to optimize during the learning process is better than optimizing fixed dimensions. However, this involves choosing the right multi-armed bandit variant. In our result, UCB was the best one.

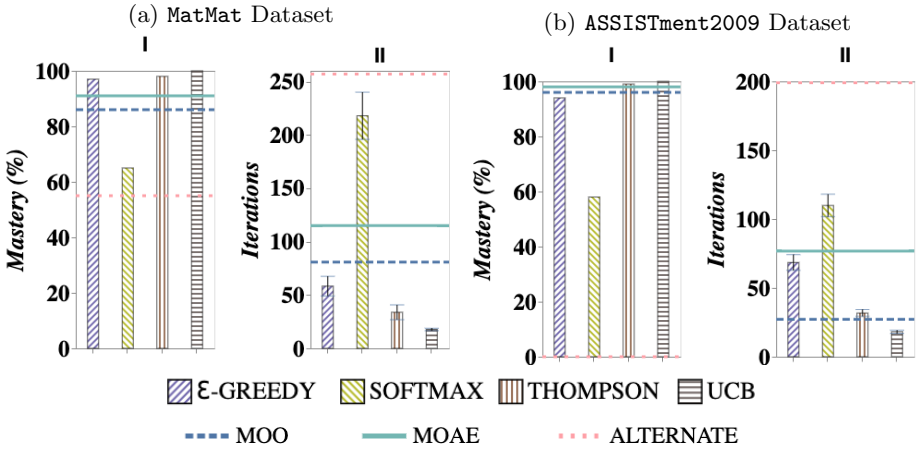


Fig. 17: **(I)** Percentage of mastery attained - **(II)** Average number of iterations to attain mastery using MAB strategies.

Response time. Time experiments have shown that MAB strategies are faster than MOO to generate the test batches but are slower than bi-objective variants. The reason is that, during the whole learning process, MAB strategies optimizes fewer objectives than MOO. In some iterations, MAB leverages only two objectives. So they are faster in generating the test batches. On the other hand, MAB strategies optimize more objectives than bi-objective variants. In some iterations, MAB optimizes for all objectives. So they are slower in generating the batches.

Insights on Combining Dimensions. One can see that bi-objective and multi-objective variants are a special case of a MAB strategy where only one arm is available and chosen. Based on that, one can ask the question of whether the policies of MAB are relying on only one or two variants, for example, they leverage both best variants MOO and MOAE. To answer that, we examine the policies output by MAB on MatMat.

First, we examine the overall proportions of the selection of each variant in each strategy. The results show that the best strategies (the ones exhibiting the highest mastery rates in lower iterations) UCB and THOMPSON, exhibit a more uniform use of each variant. For example, in UCB each multi-objective variant is

selected ≈ 25% of the time. In contrast, we see that SOFTMAX, the worst strategy, relies mainly on two variants, MOEG with ≈ 84% and MOAE with ≈ 13% of the time. This may be the reason for its underperformance. Another interesting insight is that ϵ-GREEDY selects MOO just 9% of the time. This also explains why ϵ-GREEDY has a higher number of iterations and a slightly slower skill progression.

Analyzing these proportions in more detail showed that UCB is more stable and less noisy in selecting the different variants across all simulations. For example, by calculating the standard deviation of MOO selection proportions we found that UCB has the lowest value (≈ 0.04) while SOFTMAX has the highest one (≈ 0.4). This means that the choice of MOO in UCB is similar from one simulation to another while for SOFTMAX this choice looks more random and noisier.

We now examine the veracity of the hypotheses we made in Section 3.2. We assumed that after failing tests, it is more desirable to optimize gap. We also assumed that after obtaining successful answers, aptitude is optimized. Our results show that all strategies tend to leverage gap, in the next two iterations, after learners fail to increase their skill value. For example, UCB and THOMPSON optimize gap 77% and 72% of the time after wrong answers while SOFTMAX does the same 67% of the time. Similarly, our simulations show that UCB, THOMPSON, and ϵ-GREEDY optimize aptitude after successful tests more than 75% of the time, while it is no more than 58% for SOFTMAX. These results also provide insights on why SOFTMAX under-performs compared to the other strategies.

Findings. This experiment finds that choosing automatically the dimensions to optimize at each iteration improves the rate of mastery and the number of iterations needed to achieve it. This justifies the use of a meta-strategy to learn the best combination of objectives to optimize at each iteration.

4.9 RQ5. The quality of the generated test batches

In this section, we want to verify whether our behavioral dimensions: Expected Performance, Aptitude, and Gap, capture well the needs of real learners. We aim to test if these dimensions define well the knowledge of the learners. We then compare our solution, MOO, to the state-of-the-art adaptive learning solutions [2] presented in Section 4.4. We used two of the datasets presented in Section 4.1: ASSISTment and ASSISTment2009. In the following, we first present the settings of the experiments and then report the comparison results.

Relevance of assigned batches. Tables 3 and 4 report the overall average of Precision, Recall, and F-score over all learners on ASSISTment and ASSISTment2009 respectively. The main observation is that our variants MOO and MOO_BEG are the best solutions for relevance. For example, the Precision of MOO_BEG and MOO is 10% and 9.2% on ASSISTment respectively while the best baseline Precision is 8.8% (i.e., IRT). This means that our solution tends to sufficiently capture the knowledge of the learners and trace its evolution better than all the baselines. More precisely, one can see that MOO_BEG outperforms MOO on both datasets. The

reason is that MOO is more restrictive as it only assigns tests that have a higher difficulty than the current skill level of the learner. This shows that, in the real world, learners tend to come back and complete previously assigned tests even if they were mastered. For example, on ASSISTment, 41% of the tests assigned by MOO_BEG were already mastered by learners while this rate is null for MOO.

The second observation is that deep knowledge models tend to have better performances than traditional ones (e.g., BKT, IRT, and MCD). More precisely, we observe that SAKT is the best baseline on ASSISTment2009 and the second best on ASSISTment. It also outperforms all other deep knowledge models. The reason is that it relies on the powerful attention mechanism which mimics cognitive attention and learns the importance and relevance of tests from the input sequence (i.e., the learning process). Our results confirm that the attention mechanism outperforms recurrent models, in an upskilling environment, as it was originally introduced [61] to overcome the drawback of recurrent models (e.g., LSTM [25]). Indeed, recurrent models lose information when embedding the interactions of a long input sequence. Our results also show that models used originally for recommendations can be applied for upskilling. Indeed, we see that KNN has reasonable relevance as it outperforms models like BKT or NCDM on both datasets.

Finally, one can see a drop in performance from ASSISTment to ASSISTment2009. The reason is directly related to the size of the data. Indeed, ASSISTment2009 has more than 27600 tests while ASSISTment contains around 3000 tests. Most importantly, we see that our solutions MOO and MOO_BEG scale well when the number of tests increases.

Knowledge of assigned batches. Figure 18 shows the percentage of learners that improved their skill in the next iteration as a function of the average skill (i.e., knowledge) that was gained by these learners on both ASSISTment and ASSISTment2009 datasets. These results are generated by using BKT as a simulator for each learner. From Figure 18 (a), one can see that our solutions MOO and MOO_BEG are outperformed by knowledge tracing baselines in terms of the percentage of learners that improved their skill. One potential explanation for these results is that the knowledge models tend to find a better connection between the embedding of the tests and the performances of the learners. However, despite that, our solutions are the best in terms of the knowledge that these learners gained. We explain this outperformance for the skill gain by the fact that our solution always challenges the learners by maximizing Aptitude while all other baselines tend to optimize only for the Expected Performance. Finally, we see that KNN is the worst performer for both metrics which indicates that recommendation-based solutions tend to assign tests that match the previous sequence the learners interacted with instead of the ones that improve their knowledge.

Models		Precision (%)	Recall (%)	F-Score (%)
Recommendation Based	KNN	5.2	1.128	1.854
Traditional Knowledge Tracing	BKT	1	0.232	0.376
	IRT	8.8	1.734	2.897
	MCD	2.8	0.573	0.951
Deep Knowledge Tracing	NCDM	5	1.051	1.737
	DKT	5.2	1.123	1.847
	DKVMN	4.8	0.908	1.527
	SAKT	6.8	1.333	2.229
Ours	MOO	9.2	2.074	3.385
	MOO_BEG	**10**	**2.151**	**3.54**

Table 3: Overall relevance results on ASSISTment dataset

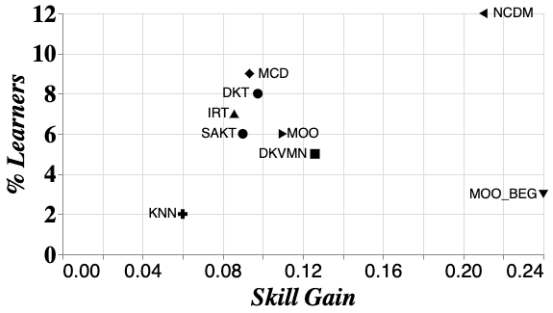
Models		Precision (%)	Recall (%)	F-Score (%)
Recommendation Based	KNN	0.4	0.01	0.02
Traditional Knowledge Tracing	BKT	0	0	0
	IRT	0.6	0.014	0.027
	MCD	1	0.021	0.042
Deep Knowledge Tracing	NCDM	0	0	0
	DKT	0.6	0.014	0.028
	DKVMN	1	0.026	0.051
	SAKT	1.4	0.037	0.072
Ours	MOO	1	0.024	0.048
	MOO_BEG	**2.2**	**0.055**	**0.107**

Table 4: Overall relevance results on ASSISTment2009 dataset

On larger datasets (i.e., ASSISTment2009), we see that NCDM is the worst baseline (Figure 18 (b)) while it was the best in the previous data. This baseline does not scale when the number of tests increases. However, we see that our solutions MOO and MOO_BEG are the best for both metrics with an advantage for the former one (e.g., ≈20% of learners improved their skills with an average gain of 0.23). This shows that our methods scale well to the size of the data. Finally, we see that all knowledge tracing baselines have similar performances as was the case in the previous dataset.

Findings. This experiment we performed in this part finds that our solutions are not only the best in terms of skill gain but it offers the possibility to a larger number of learners to improve their knowledge. In addition to that, they assign more relevant test batches according to the previous sequence of learners' interactions. Finally, in this experiment, we found that our solution scales for larger datasets. In conclusion, the dimensions, we defined in this work, tend to capture well the knowledge of the learners and are able to trace their evolution.

(a) `ASSISTment` Dataset



(b) `ASSISTment2009` Dataset

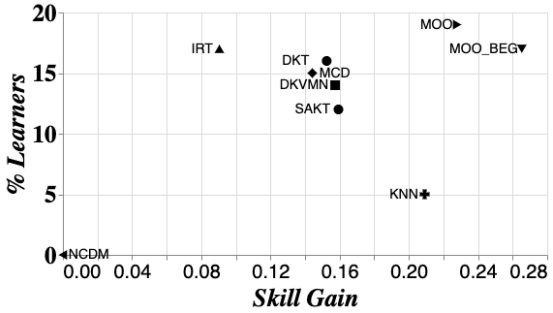


Fig. 18: Average skill gain and average rate of learners who improved their skills for all baselines.

5 Related work

Education Science. Flow [15] and ZPD [62] theories conceptualize the idea of *experiential learning* [28] 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 [38, 18]. More recently, it was used in crowdsourcing to compose tasks with different difficulty levels and test the impact on skill improvement and worker satisfaction [36]. *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 [49] develop criteria that analyze the sequential interactions of learners and determine if a learner has mastered a skill. The easiest methods are based on simple statistics. For example, *NCC* (*N Consecutive Correct*) [27] declares mastery if the number of consecutive correct answers exceeds a certain threshold. Another method is *Moving Average* [50] that declares mastery of a type of test if the average of correct answers within a moving window exceeds a threshold. These methods are

too simple to capture learners' knowledge of all the skills. For this reason, more sophisticated models were also proposed [2, 3]. The two first and most popular models are *Bayesian Knowledge Tracing (BKT)* [14] and *Latent Factor* models [11, 48, 50].

BKT [14] 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 [16, 43, 2]. For example, KT-IDEM [47] accounts for the difficulty of each test. Other improvements are made by applying new architecture like Deep Learning techniques. The first extension is Deep Knowledge Tracing (DKT) [51] which uses Recurrent Neural Networks (RNNs) [32], and more precisely LSTM [25], to capture the past performance of the learners chronologically. Other models used key-value memory networks [37, 1] to capture these interactions. More models used the concept of BKT with specific data types like text [57] or graphs [41].

On the other hand, *Latent Factor* models are based on logistic regression [11]. They define the performances of the learners using latent parameters and learn them to infer the probability of mastery using a sigmoid function. The most known model is Item Response Theory (IRT) [53] which ignores the chronological aspect of the learner interaction and considers that tests are independent. Other models like Additive Factor Model (AFM) [12] or Performance Factor Analysis (PFA) [48] were proposed by considering other assumptions like the pace with which the learners master a certain skill (or difficulty a difficulty level).

In our work, we leverage some of these models to simulate learners and their answers (e.g., KT-IDEM [47] and IRT [53]). We also used some of these models as baselines to compare the quality of the assigned batches of tests (e.g., DKT [51], and memory networks [37]). Finally, the simplest statistical methods (e.g., NCC) was used in our skill update strategy.

Adaptive Learning. Adaptive learning systems aim to provide an efficient, effective, and customized learning experience for learners by capturing their competencies and interactions with various learning activities and dynamically adapting learning content to suit their individual abilities or preferences [60]. A consistent and growing body of knowledge provides evidence about the effectiveness of adaptive systems compared to classroom teaching or to educational systems that provide instructions and learning activities that are not adaptive [65]. While there are examples of using adaptive learning systems across different disciplines, by and large, they have been most effectively utilized in the context of high school Maths using tools such as ASSISTments [24]. Usually, learner modeling models and knowledge tracing were used for adaptive learning and test assignment [2]. Recent work also combines different types of data and deep learning architectures for adaptive learning [26, 33]. In [26], the authors propose

to learn a policy based on Reinforcement Learning [4] that optimizes different objectives: Engagement of the learner, Smoothness of the tests, and the trade-off between exploration and exploitation of new tests. The proposed policy outperformed standard baselines like IRT [53], or DKT [51]. Another work by [33] integrates both DKT as a knowledge tracing and a graph-based model as a cognitive navigation for learning path assignment. Their procedure outperforms recommendation-based and simple knowledge models. Finally, [40] also proposes a bi-objective solution for test assignment. The solution maximizes the precision of the assigned tests while minimizing the number of given tests. Their simulated experiments showed the effectiveness of the proposed solution in maximizing the accuracy while minimizing the size of the given tests.

Similarly to some of these works, we aim to solve a multi-objective problem for adaptive learning. On the contrary, and to the best of our knowledge, none of these solutions combines the optimization of expected performance, aptitude, and skill gap to adapt tests to individual learners.

In online labor marketplaces, a few studies focused on the role of task difficulty and workers' ability to complete micro-tasks in improving skills [21], and how affinity between workers can be used to form teams that collaborate to produce high-quality contributions while also improving skills [19].

Usually, such approaches require additional human costs to build training material or give feedback to workers. Additionally, these solutions do not customize test difficulty in recommended tasks.

6 Conclusion and Future Work

We addressed 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. We proposed two approaches: `MOO` that directly solves our problem and a `MAB` that chooses among different optimization variants at each iteration. We tested the impact of optimizing these dimensions on skill progression and mastery achievement. We also tested the impact of different learner simulation models on mastery achievement. Our experiments confirmed that `MAB` offers a higher mastery rate and a better final skill gain than `MOO`. They also confirmed that our solution `MOO` assigns tests with higher quality and accuracy.

For future work, we would like to deploy our solutions so that real learners can interact with them. We may use environments at our university that scaffolds students' activity as they learn to solve exercises or write experimental reports. Experimenting online with real learners will help confirm the findings we exhibit in this work. It will also permit to capture new variables as completion time, reflexion time, or non cognitive metrics (e.g., engagement, motivation) [20].

In addition to that, we aim to extend our formalization by considering additional theories. In fact, there are many learning theories in the physical world, such as situated learning [29] and collaborative learning [10]. One representa-

tive of the former is apprenticeship where knowledge is propagated from experts to novice learners based on the principle of *Legitimate Peripheral Participation* [29]. 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 [17, 13, 64].

We also aim to personalize the upskilling experience of learners. We would like to model profiles for learners based on their past performances on the tests of different skills. We may use these profiles to assign tests by either using a clustering method to define their overall ability [39] or applying a collaborative filtering method [56] to focus on the tests that were correctly completed by similar learners.

References

1. Abdelrahman, G., Wang, Q.: Knowledge tracing with sequential key-value memory networks. In: Proceedings of the 42nd international ACM SIGIR conference on research and development in information retrieval. pp. 175–184 (2019)
2. Abdelrahman, G., Wang, Q., Nunes, B.: Knowledge tracing: A survey. ACM Computing Surveys **55**(11), 1–37 (2023)
3. Abdi, S., Khosravi, H., Sadiq, S., Darvishi, A.: Open learner models for multi-activity educational systems. In: Artificial Intelligence in Education: 22nd International Conference, AIED 2021, Utrecht, The Netherlands, June 14–18, 2021, Proceedings, Part II. pp. 11–17. Springer (2021)
4. Arulkumaran, K., Deisenroth, M.P., Brundage, M., Bharath, A.A.: Deep reinforcement learning: A brief survey. IEEE Signal Processing Magazine **34**(6), 26–38 (2017)
5. Badrinath, A., Wang, F., Pardos, Z.: pybkt: An accessible python library of bayesian knowledge tracing models. In: Proceedings of the 14th International Conference on Educational Data Mining. pp. 468–474 (2021)
6. Bartolini, I., Ciaccia, P., Patella, M.: Efficient sort-based skyline evaluation. ACM Transactions on Database Systems (TODS) **33**(4), 1–49 (2008)
7. Basawapatna, A.R., Repenning, A., Koh, K.H., Nickerson, H.: The zones of proximal flow: guiding students through a space of computational thinking skills and challenges. In: Proceedings of the ninth annual international ACM conference on International computing education research. pp. 67–74 (2013)
8. Birnbaum, A.: Some latent trait models and their use in inferring an examinee's ability. Statistical theories of mental test scores (1968)
9. Bouarour, N., Benouaret, I., D'Ham, C., Amer-Yahia, S.: Adaptive test recommendation for mastery learning. In: Proceedings of the 2nd International Workshop on Data Systems Education: Bridging education practice with education research. pp. 18–23 (2023)
10. Bruffee, K.A.: Collaborative learning: Higher education, interdependence, and the authority of knowledge. ERIC (1999)
11. Cen, H., Koedinger, K., Junker, B.: Learning factors analysis–a general method for cognitive model evaluation and improvement. In: Intelligent Tutoring Systems: 8th International Conference, ITS 2006, Jhongli, Taiwan, June 26-30, 2006. Proceedings 8. pp. 164–175. Springer (2006)

12. Cen, H., Koedinger, K., Junker, B.: Comparing two irt models for conjunctive skills. In: Intelligent Tutoring Systems: 9th International Conference, ITS 2008, Montreal, Canada, June 23-27, 2008 Proceedings 9. pp. 796–798. Springer (2008)
13. Coetzee, D., Lim, S., Fox, A., Hartmann, B., Hearst, M.A.: Structuring interactions for large-scale synchronous peer learning. In: Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing. pp. 1139–1152 (2015)
14. Corbett, A.T., Anderson, J.R.: Knowledge tracing: Modeling the acquisition of procedural knowledge. User modeling and user-adapted interaction **4**(4), 253–278 (1994)
15. Csikszentmihalyi, M.: Beyond boredom and anxiety: The experience of play in work and games. Jossey-Bass (1975)
16. David, Y.B., Segal, A., Gal, Y.: Sequencing educational content in classrooms using bayesian knowledge tracing. In: Proceedings of the sixth international conference on Learning Analytics & Knowledge. pp. 354–363 (2016)
17. Davis, D., Chen, G., Hauff, C., Houben, G.J.: Activating learning at scale: A review of innovations in online learning strategies. Computers & Education **125**, 327–344 (2018)
18. De Vin, L.J., Jacobsson, L., Odhe, J., Wickberg, A.: Lean production training for the manufacturing industry: Experiences from karlstad lean factory. Procedia Manufacturing **11**, 1019–1026 (2017)
19. Esfandiari, M., Wei, D., Amer-Yahia, S., Basu Roy, S.: Optimizing peer learning in online groups with affinities. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. pp. 1216–1226 (2019)
20. Fischer, C., Pardos, Z.A., Baker, R.S., Williams, J.J., Smyth, P., Yu, R., Slater, S., Baker, R., Warschauer, M.: Mining big data in education: Affordances and challenges. Review of Research in Education **44**(1), 130–160 (2020)
21. Gadiraju, U., Dietze, S.: Improving learning through achievement priming in crowdsourced information finding microtasks. In: Proceedings of the Seventh International Learning Analytics & Knowledge Conference. pp. 105–114. ACM (2017)
22. Gašević, D., Kovanović, V., Joksimović, S., Siemens, G.: Where is research on massive open online courses headed? a data analysis of the mooc research initiative. International Review of Research in Open and Distributed Learning **15**(5), 134–176 (2014)
23. Gong, Y., Beck, J.E., Heffernan, N.T.: Comparing knowledge tracing and performance factor analysis by using multiple model fitting. ITS2010 Intelligent Tutoring Systems. LNCS **6094**, 35–44 (2010)
24. Heffernan, N.T., Heffernan, C.L.: The assistments ecosystem: Building a platform that brings scientists and teachers together for minimally invasive research on human learning and teaching. International Journal of Artificial Intelligence in Education **24**(4), 470–497 (2014)
25. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997)
26. Huang, Z., Liu, Q., Zhai, C., Yin, Y., Chen, E., Gao, W., Hu, G.: Exploring multi-objective exercise recommendations in online education systems. In: Proceedings of the 28th ACM International Conference on Information and Knowledge Management. pp. 1261–1270 (2019)
27. Kelly, K., Wang, Y., Thompson, T., Heffernan, N.: Defining mastery: Knowledge tracing versus n-consecutive correct responses. Student Modeling From Different Aspects p. 39 (2016)

28. Kolb, D.A.: Experiential learning: Experience as the source of learning and development. Englewood Cliffs (1984)
29. Lave, J., Wenger, E.: Situated learning: Legitimate peripheral participation. Cambridge university press (1991)
30. Lee, Y.W., Sawaki, Y.: Cognitive diagnosis approaches to language assessment: An overview. Language Assessment Quarterly **6**(3), 172–189 (2009)
31. Li, L.: Reskilling and upskilling the future-ready workforce for industry 4.0 and beyond. Information Systems Frontiers pp. 1–16 (2022)
32. Lipton, Z.C., Berkowitz, J., Elkan, C.: A critical review of recurrent neural networks for sequence learning. arXiv preprint arXiv:1506.00019 (2015)
33. Liu, Q., Tong, S., Liu, C., Zhao, H., Chen, E., Ma, H., Wang, S.: Exploiting cognitive structure for adaptive learning. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. pp. 627–635 (2019)
34. Lord, F.M.: Applications of item response theory to practical testing problems. Routledge (1980)
35. Mao, Y., Xu, B., Yu, J., Fang, Y., Yuan, J., Li, J., Hou, L.: Learning behavior-aware cognitive diagnosis for online education systems. In: International Conference of Pioneering Computer Scientists, Engineers and Educators. pp. 385–398. Springer (2021)
36. Matsubara, M., Borromeo, R.M., Amer-Yahia, S., Morishima, A.: Task assignment strategies for crowd worker ability improvement. Proc. ACM Hum. Comput. Interact. **5**(CSCW2), 1–20 (2021)
37. Miller, A., Fisch, A., Dodge, J., Karimi, A.H., Bordes, A., Weston, J.: Key-value memory networks for directly reading documents. arXiv preprint arXiv:1606.03126 (2016)
38. Mincer, J.: On-the-job training: Costs, returns, and some implications. Journal of political Economy **70**(5, Part 2), 50–79 (1962)
39. Minn, S.: Bkt-lstm: Efficient student modeling for knowledge tracing and student performance prediction. arXiv preprint arXiv:2012.12218 (2020)
40. Mujtaba, D.F., Mahapatra, N.R.: Multi-objective optimization of item selection in computerized adaptive testing. In: Proceedings of the Genetic and Evolutionary Computation Conference. pp. 1018–1026 (2021)
41. Nakagawa, H., Iwasawa, Y., Matsuo, Y.: Graph-based knowledge tracing: modeling student proficiency using graph neural network. In: IEEE/WIC/ACM International Conference on Web Intelligence. pp. 156–163 (2019)
42. Omidvar-Tehrani, B., Amer-Yahia, S., Dutot, P.F., Trystram, D.: Multi-objective group discovery on the social web. In: ECML/PKDD. pp. 296–312. Springer (2016)
43. Ostrow, K., Donnelly, C., Adjei, S., Heffernan, N.: Improving student modeling through partial credit and problem difficulty. In: Proceedings of the Second (2015) ACM Conference on Learning@ Scale. pp. 11–20 (2015)
44. Pandey, S., Karypis, G.: A self-attentive model for knowledge tracing. arXiv preprint arXiv:1907.06837 (2019)
45. Panel, H.E.S.: Final report–improving retention, completion and success in higher education (2017)
46. Papadimitriou, C.H., Yannakakis, M.: On the approximability of trade-offs and optimal access of web sources. In: Proceedings 41st annual symposium on foundations of computer science. pp. 86–92. IEEE (2000)
47. Pardos, Z.A., Heffernan, N.T.: Kt-idem: Introducing item difficulty to the knowledge tracing model. In: International conference on user modeling, adaptation, and personalization. pp. 243–254. Springer (2011)

48. Pavlik Jr, P.I., Cen, H., Koedinger, K.R.: Performance factors analysis–a new alternative to knowledge tracing. Online Submission (2009)
49. Pelánek, R., Řihák, J.: Experimental analysis of mastery learning criteria. In: Proceedings of the 25th Conference on User Modeling, Adaptation and Personalization. pp. 156–163 (2017)
50. Pelánek, R.: Application of time decay functions and elo system in student modeling. Proc. of Educational Data Mining pp. 21–27 (01 2014)
51. Piech, C., Bassen, J., Huang, J., Ganguli, S., Sahami, M., Guibas, L.J., Sohl-Dickstein, J.: Deep knowledge tracing. Advances in neural information processing systems **28** (2015)
52. Rasch, G.: Probabilistic models for some intelligence and attainment tests. ERIC (1993)
53. Reckase, M.D., Reckase, M.D.: Unidimensional item response theory models. Multidimensional item response theory pp. 11–55 (2009)
54. Rollinson, J., Brunskill, E.: From predictive models to instructional policies. International Educational Data Mining Society (2015)
55. Rozenblit, L., Keil, F.: The misunderstood limits of folk science: An illusion of explanatory depth. Cognitive science **26**(5), 521–562 (2002)
56. Sarwar, B., Karypis, G., Konstan, J., Riedl, J.: Item-based collaborative filtering recommendation algorithms. In: Proceedings of the 10th international conference on World Wide Web. pp. 285–295 (2001)
57. Su, Y., Liu, Q., Liu, Q., Huang, Z., Yin, Y., Chen, E., Ding, C., Wei, S., Hu, G.: Exercise-enhanced sequential modeling for student performance prediction. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 32 (2018)
58. Sutton, R.S., Barto, A.G.: Reinforcement learning: An introduction. MIT press (2018)
59. bigdata ustc: Educdm. https://github.com/bigdata-ustc/EduCDM (2021)
60. VanLehn, K.: The relative effectiveness of human tutoring, intelligent tutoring systems, and other tutoring systems. Educational psychologist **46**(4), 197–221 (2011)
61. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017)
62. Vygotsky, L.: Zone of proximal development. Mind in society: The development of higher psychological processes **5291**, 157 (1987)
63. Wang, F., Liu, Q., Chen, E., Huang, Z., Chen, Y., Yin, Y., Huang, Z., Wang, S.: Neural cognitive diagnosis for intelligent education systems. In: Proceedings of the AAAI conference on artificial intelligence. vol. 34, pp. 6153–6161 (2020)
64. Wu, A.S., Farrell, R., Singley, M.K.: Scaffolding group learning in a collaborative networked environment (2002)
65. Xu, Z., Wijekumar, K., Ramirez, G., Hu, X., Irey, R.: The effectiveness of intelligent tutoring systems on k-12 students' reading comprehension: A meta-analysis. British Journal of Educational Technology **50**(6), 3119–3137 (2019)
66. Zhang, J., Shi, X., King, I., Yeung, D.Y.: Dynamic key-value memory networks for knowledge tracing. In: Proceedings of the 26th international conference on World Wide Web. pp. 765–774 (2017)
67. Zhu, H., Tian, F., Wu, K., Shah, N., Chen, Y., Ni, Y., Zhang, X., Chao, K.M., Zheng, Q.: A multi-constraint learning path recommendation algorithm based on knowledge map. Knowledge-Based Systems **143**, 102–114 (2018)