Fast Weakness Identification for Adaptive Feedback

Raymond Morland[1](✉), Lawrence Wang[2], and Fuhua Lin[1]

[1] Athabasca University, Athabasca, AB T9S 3A3, Canada
rmorland1@learn.athabascau.ca, oscarl@athabascau.ca
[2] University of Alberta, Edmonton, AB T6G 2R3, Canada
lxwang@ualberta.ca

Abstract. Identifying and addressing areas of weakness of online learning students early on is critically important to prevent minor issues from becoming major obstacles to their success. It is desirable to have a tool that allows learners to conduct personalized formative assessment on demand anytime during their course study. To minimize the cognitive load of a learner and facilitate the iterative learning process, a pedagogical strategy is to identify a singular weak skill each formative assessment and to provide adaptive feedback for remediation for the learner to close the gap between his/her current performance and the expected mastery criteria. As one gap closes, another gap may be identified afterward, renewing the formative assessment and feedback loop. For such singular weakness identification, minimizing the time spent or the number of questions on each assessment is crucial for maintaining learner engagement. On the other hand, it is also critical to ensure that the result of the assessment is reliable to provide effective feedback. To balance the accuracy and efficiency of the assessment, we propose three algorithms for fast and adaptive weakness identification based on the good arm identification (GAI) problem in multi-armed bandit-based machine learning. We evaluate the sensitivity and performance of the proposed algorithms through simulation.

Keywords: Adaptive Learning · Formative Assessment · Multi-armed Bandits · Simulated Learners · Knowledge Tracing · Knowledge Components

1 Introduction

Formative assessments are essential in the educational process, as they provide timely opportunities for students to close the gap between their current performance and the expected criteria. By identifying and addressing areas of weakness early on, students can prevent minor issues from becoming major obstacles to their success. However, most existing research for conducting formative assessment is for classroom-based educational environments, in which the most efficient way of gathering evidence about student weaknesses and taking actions for improvement is face-to-face interaction between teachers and students [1]. In online education, especially asynchronous online learning (AOL) or Self-directed learning (SDL), such face-to-face discussions are infeasible. In AOL, feedback on curriculum-embedded assignments, quizzes, and projects may not

A. Sifaleras and F. Lin (Eds.): ITS 2024, LNCS 14798, pp. 40–54, 2024.
https://doi.org/10.1007/978-3-031-63028-6_4

be as immediate as in traditional classroom settings. The delay in receiving feedback can hinder the learning process, as students may move on to new topics without fully understanding or correcting mistakes from previous lessons. Also, providing personalized feedback to each student can be challenging for instructors, especially in courses with many participants. In SDL, self-assessment is common, and external feedback is limited unless the learner seeks it out from peers, mentors, or through other means. In adaptive learning systems (ALS), it is pivotal to generate immediate feedback and formative assessments that are integrated into the learning path, using these inputs to adjust the learning experience in real time.

Thus, automated formative assessment and feedback mechanisms are desirable for online learning. However, to be helpful, they should address specific student needs or misunderstandings as effectively as personalized, instructor-led feedback. Gareis (2007) specifies two criteria to ensure formative assessment is beneficial for the student [2]. First, it must convey the student's progress in learning the material. Second, it must specify steps to continue learning. Therefore, by identifying a student's learning weaknesses and strengths, formative assessments can help identify the next steps the student should take in their learning path. Furthermore, to be maximally beneficial, formative assessments should be made available anytime and on-demand during a course. To make this possible, formative assessments can be administered using intelligent tutoring systems.

Intelligent tutoring systems can generate automated feedback that reduces discrepancies between current and desired performance. To generate effective feedback, it is crucial to focus on how well the task is being performed [3]. Given a set of knowledge components (KCs) (or learning objectives or skills), formative assessment is performed by asking a set of questions to the learners and analyzing the responses of the learner. Depending on the pedagogical strategy the teacher prefers, a formative assessment can terminate once one or several "weak" or "un-mastered" KCs or the weakest KCs are detected, and the remediation feedback can be generated by further investigating about the foremost weak KCs using the chain of weakness [4].

In formative assessment, minimizing the time spent or the number of questions on each formative assessment is crucial for maintaining engagement [5], reducing cognitive load, providing timely and frequent feedback, encouraging reflection, supporting personalized learning, enhancing motivation, and ensuring efficient use of time. These elements collectively contribute to a more effective and engaging learning experience. On the other hand, it is also critical to ensure that formative assessment is reliable to generate feedback. Therefore, there is a need for balancing the accuracy and efficiency of the assessment. To meet this need, in this paper, we model a formative assessment process as a *good arm identification* (GAI) problem in machine learning [6] and propose three fast and adaptive weakness identification algorithms. The sensitivity and performance of the proposed algorithms is verified by simulation.

The rest of the paper is organized as follows. Related work is reviewed in Sect. 2. Section 3 formulates the weakness identification problem and the proposed algorithms to solve the problem. Next, our simulation setup, experimental results and discussions are presented in Sect. 4. Finally, we conclude the paper and point out future research directions in Sect. 5.

2 Related Work

Our approach for adaptive feedback generation is involved with knowledge tracing in intelligent tutoring systems, and bandit algorithms in machine learning. Efficiently identifying the exact learning items that would prove most useful to a student's study can be classified as a Knowledge Tracing (KT) problem. KT research has been studied in similar intelligent tutoring systems under a broad range of methods to effectively track the learning progress of a student [7]. These models rely on dynamically updating student knowledge states based on their responses to provide personalized feedback and adaptivity, which may not be as relevant in scenarios in formative assessment where student mastery does not evolve.

Several innovative approaches to adaptive testing have been developed to quickly identify students' weaknesses. Kingsbury and Houser (2008) introduced ICAT: An Adaptive Testing Procedure, which is designed to efficiently identify students' areas of strength and weakness compared to traditional adaptive testing methods [8]. Yigit, Sorrel, and de la Torre (2018) discuss using the Jensen–Shannon divergence index in CD-CAT to improve attribute classification accuracy with very short test lengths, allowing for quick identification of students' weaknesses [9]. These approaches represent a significant advancement in adaptive testing, offering more personalized, efficient, and accurate methods for identifying and addressing students' weaknesses. However, their algorithms can be enhanced to better handle uncertainty in student responses. In this research, we use techniques from machine learning to deal with ambiguities in student data and make more nuanced inferences about their knowledge states.

Multi-armed bandits are a reinforcement learning mechanism that traditionally aims to maximize the attained reward from a set of items with varying, unknown reward distributions [10]. The rewards of these items, referred to as arms, can be probability distributions. The arms, in our context, refer to the KCs in a domain and the rewards of the arms mean the proficiencies or knowledge states of a learner. One heuristic for choosing arms that addresses the exploration-exploitation dilemma in the multi-armed bandit problem is Thompson Sampling [10], which may use beta distributions to characterize the state of each arm. The beta distribution is handy for representing knowledge about probabilities [11]. Our model adopts Beta distributions to maintain empirical observations.

Kano et al. consider a novel stochastic multi-armed bandit problem called *good arm identification* (GAI) [6], where a good arm is defined as an arm with expected reward greater than or equal to a given threshold. They proposed an algorithm to solve the *exploration-exploitation dilemma of confidence* that GAI faces. In the context of weakness identification through adaptive assessment, the standard bandit model of exploration-exploitation does not accurately apply as we are only concerned with the identification of a weak arm rather than best arm identification as best arm identification is much more time-consuming [4, 6]. Therefore, in this paper we model our problem as *good-arm identification* modified to identify *weak* arms. This approach is purely explorative in nature and aims to quickly identify any such arm(s) less than a specified threshold.

Another multi-armed bandit framework relevant to our case is *multi-armed bandits with correlated arms* proposed by Gupta et al. [12]. We use it to model dependency among knowledge components or learning objectives. As such, our algorithm borrows ideas developed for correlated arms such as the concept of *pseudo-rewards*: sampling arm k may generate reward r_1 for itself, but similar, correlated rewards r_2 may be associated with correlated arm l [12].

3 Problem Formulation

The goal of formative assessment is to identify and close the gap between the status of student learning and the desired learning objective. Placing appropriate demands on learners with the ZPD assists them to close the gap. As one gap closes, new learning goals are identified or another gap is identified, renewing the formative assessment feedback loop [1]. Figure 1 shows the process of iteratively generating feedback about remediation action, in which a singular weak skill each formative assessment is identified.

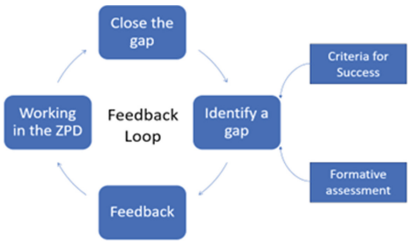

Fig. 1. The process of iteratively generating feedback about remediation action, in which a singular weak skill each formative assessment is identified.

3.1 Domain Model

The goal of our research is to allow educators and students to efficiently identify their critical weaknesses in a particular subject. The set of all topics that the subject consists of and their relationships to each other can be described by a domain model. The domain model consists of a set of knowledge components or learning objectives, hereafter referred to as KCs, modelled as a graph-like structure where the nodes represent the KC, and the edges represent the dependencies between them. We can use the dependencies of the KCs to our advantage; i.e., if a KC is dependent on another, we can extract information from both simply by sampling either one. Using this KC structure, we can approach the problem of identifying weak KCs as a multi-armed bandit "good arm identification" problem.

3.2 Student Model

To provide appropriate question sequencing, a student model of the learners' abilities is needed. For a student s learning within a domain consisting of K knowledge components: each knowledge component $i \in \{1, 2, \ldots, K\}$ is associated with their mastery in i. This mastery level represents the skill level of student s in KC i. In our Beta-Bernoulli MAB, a student's abilities, and history in KC i are modelled as a beta distribution with shape parameters (α_i, β_i) and values (μ_i, σ_i^2). The student's mastery of KC i is represented by the mean, μ_i, of this distribution. Therefore, the mean represents the probability that the student will respond correctly to a question from KC i. The shape parameters represent the number of questions the student has answered correctly, α_i, and incorrectly, β_i, for KC i.

3.3 Transition Model

The transition model in an adaptive learning system maps an individual learner and their current knowledge state to their next logical knowledge state. In adaptive formative assessment, an instructor can specify the following parameters: mastery threshold and confidence level, or error rate. To define what makes an arm weak, we instate a mastery threshold ξ as a hyperparameter. More generally, the instructor can specify a mastery threshold denoted as ξ_i for each KC or arm i. If our empirical mean reward μ_i for arm i is below ξ_i, that is, $\mu_i < \xi_i$, we can define it as weak.

3.4 Bandit Model

The standard MAB model cannot be directly used to model the weakness identification problem in formative assessment since we are simply interested in the identification of a weak arm, not the exploitation of rewards. Also, even though identifying the weakest skill or KC would be ideal, identifying the best arm requires drawing many arms. In our problem, the arms of the MAB are represented by KCs. A bad arm is also called a weak arm or a weak KC. The system sequentially chooses KCs, selects questions from those KCs, and observes independent noisy responses to the questions by the student. If the student answers a question correctly, the corresponding arm, or KC, gets a reward. This means updating the student model's beta distribution parameters for that KC based on the correctness of the answer. The goal for the MAB is to select questions so that a weak KC for the student can be identified confidently with as few questions as possible.

Sampling Policies. We call the procedure of choosing the next arm to pull the sampling policy of the MAB. The sampling policy is responsible for determining the next arm to pull to minimize the number of questions to ask a student. To do this we want the sampling policy to find the current estimated weak KC so that we can find an actual weak KC quicker. This is central to the bad, or weak, arm identification problem. In this paper we explore three alternative sampling policies: High Degree of Confidence (HDoC), Thompson, and Random. The HDoC policy is modified from the HDoC algorithm introduced by Kano et al. [6]. The following function determines the score,

$\widetilde{\mu}_i(t)$, of KC i for round t:

$$\tilde{\mu}_i(t) = \hat{\mu}_i(t) + \sqrt{\frac{\log t}{2N_i(t)}} \tag{1}$$

Where $N_i(t)$ is the number of times KC i has been sampled as of round t and $\widehat{\mu}_i(t)$ is the empirical mean of the KC at round t. The arm with the lowest score is sampled so that if it is weak, it can be identified quickly.

The Thompson sampling policy we implement follows the traditional Thompson sampling algorithm for MABs [10]. During each sampling round, the beta distributions of each arm are sampled and the arm with the lowest sampled value is pulled. This arm is likely to have the highest probability of being weak, i.e. the lowest mean value. The random sampling policy naïvely selects a random arm and pulls it. It gives no preference based on the beta distributions of the arms.

Updating Empirical Means. After a question from a KC is presented to the student and after the student provides a response, the student model is updated based on the reward provided. This means updating the student model's beta distribution for the KC. Following our usage of beta distributions, the most practical way to update the empirical mean of each distribution to their respective posterior distribution is to update the α and β values. A correct answer yields updated parameters $(\alpha + 1, \beta)$, whereas an incorrect answer yield $(\alpha, \beta + 1)$ [11].

Stopping Criterion. For the MAB to terminate the question asking process, one of the stopping criteria must be met. All three algorithms will stop when there are no arms available to pull or when a weak arm is found. Arms are removed when they are determined to be strong. However, each algorithm implements a different procedure for determining whether an arm is weak or strong after a reward has been received. For the HDoC sampling policy we use the idea of score, like that used in the sampling policy and described by Kano et al. [6]. For any arm with distribution $Beta(\alpha, \beta)$ we have the following confidence bound scores $\overline{\mu}_i(t)$ and $\underline{\mu}_i(t)$:

$$\overline{\mu}_i(t) = \widehat{\mu}_i(t) + \sqrt{\frac{\log(\frac{4KN_i^2(t)}{\delta})}{2N_i^2(t)}} \tag{2}$$

$$\underline{\mu}_i(t) = \hat{\mu}_i(t) - \sqrt{\frac{\log(\frac{4KN_i^2(t)}{\delta})}{2N_i^2(t)}} \tag{3}$$

Where $\widehat{\mu}_i(t)$ is the empirical mean of the beta distribution for arm i at round t, $N_i(t)$ is the number of times arm i is sampled at round t, and δ is our accepted error rate, adjusted as a hyperparameter. This scoring function will be used to determine if we have collected sufficient samples from a particular arm i to justify outputting i as weak.

For the Thompson algorithm, we employ the standard cumulative distribution function (CDF) and error rate δ to determine whether the mean of an arm i is greater than ξ within a degree of confidence. For the random algorithm we naively classify an arm

as weak if the mean of its beta distribution is less than ξ. Otherwise, we classify it as strong. The random algorithm does not have a well-defined measure of confidence about whether an arm is weak or strong. This makes it a good candidate to compare to for algorithms that do.

Pseudo-Rewards. To take advantage of the KC mapping and dependencies we implement the idea of pseudo-rewards. Correlated bandit problems have already implemented variations of this concept [12]. In these problems, arms that are highly correlated or dependent on one another allow information from both arms to be attained by sampling merely one. For example, given dependent KC i_1 and i_2, a correct answer of arm i_1 allows us to update ($\alpha_{i_1} \leftarrow \alpha_{i_1} + 1$, $\beta_{i_1} \leftarrow \beta_{i_1}$) and ($\alpha_{i_2} \leftarrow \alpha_{i_2} + \frac{1}{c}$, $\beta_{i_2} \leftarrow \beta_{i_2}$), where c is the pseudo-reward factor. In our experiments, we set $c = 2$ to emulate a soft dependency between these arms.

3.5 Weak KC Identification Algorithms

The algorithms for weak KC identification are described as follows. The graph $\boldsymbol{G} = (\boldsymbol{KC}, \boldsymbol{E})$ represents the domain model. Where $\boldsymbol{E}$ represents the dependency relations between two KCs in the domain. The set of available arms is represented by $\boldsymbol{A}$. The mastery threshold ξ, error rate δ, and pre-sample count π, are input parameters. The number of rounds the algorithm needs to find a weak KC is represented by t. The output of the algorithms is the first KC detected as weak, if any, and the number of rounds to find the weak KC t. The pseudocode for the algorithms is shown in Figs. 2, 3, and 4.

Algorithm 1. Weak KC Identification with HDoC Sampling ($\boldsymbol{G}$, $\boldsymbol{A}$, ξ, δ, π)

1: Sample each KC in $\boldsymbol{A}$ π times.
2: Initialize $t \leftarrow \pi$
3: **repeat**
4: Select KC i^* satisfying $i^* = argmin_{i \in \boldsymbol{A}}(X_i(t))$, $X_i(t) = \hat{\mu}_i(t) + \sqrt{\frac{\log t}{2N_i(t)}}$
5: Set $t \leftarrow t + 1$
6: Select random question from KC $i^*$and get response from student
7: **if** response is correct, then update $\alpha_{i^*}(t) \leftarrow \alpha_{i^*}(t) + 1$
8: **else** update $\beta_{i^*}(t) \leftarrow \beta_{i^*}(t) + 1$
9: Update dependent KCs of i^* from $\boldsymbol{G}$ with pseudo-rewards
10: **if** $\underline{\mu}_{i^*}(t) \geq \xi$ **then** delete i^* from $\boldsymbol{A}$
11: **else if** $\bar{\mu}_{i^*}(t) < \xi$ **then** output i^*as weak KC; **break**;
12: **until** $\boldsymbol{A}$=∅

Fig. 2. Weak KC Identification Algorithm using HDoC sampling

Algorithm 2. Weak KC Identification with Thompson Sampling (*G*, *A*, ξ, δ, π)

Sample each KC in A π times.
Initialize $t \leftarrow \pi$
repeat
Select KC i^* satisfying $i^* = argmin_{i \in A}(X_i(t))$, $X_i(t) = Beta(\alpha_i, \beta_i)$
Set $t \leftarrow t + 1$
Select random question from KC $i^*$and get response from student
if response is correct, then update $\alpha_{i^*}(t) \leftarrow \alpha_{i^*}(t) + 1$
else update $\beta_{i^*}(t) \leftarrow \beta_{i^*}(t) + 1$
Update dependent KCs of i^* from G with pseudo-rewards
if $1 - CDF(\xi, \alpha_{i^*}, \beta_{i^*}) \geq 1 - \delta$ **then** delete i^* from A
else if $1 - CDF(\xi, \alpha_{i^*}, \beta_{i^*}) < \delta$ **then** output i^*as weak KC; **break**;
until A=∅

Fig. 3. Weak KC Identification Algorithm using Thompson sampling

Algorithm 3. Weak KC Identification with Random Sampling (*G*, *A*, ξ, δ, π)

Sample each KC in A π times.
Initialize $t \leftarrow \pi$
repeat
Select random KC i^* from A
Set $t \leftarrow t + 1$
Select random question from KC $i^*$and get response from student
if response is correct, then update $\alpha_{i^*}(t) \leftarrow \alpha_{i^*}(t) + 1$
else update $\beta_{i^*}(t) \leftarrow \beta_{i^*}(t) + 1$
Update dependent KCs of i^* from G with pseudo-rewards
if $mean(\alpha_{i^*}, \beta_{i^*}) \geq \xi$ **then** delete i^* from A
else if $mean(\alpha_{i^*}, \beta_{i^*}) < \xi$ **then** output i^*as weak KC; **break**;
until A=∅

Fig. 4. Weak KC Identification Algorithm using Random sampling

4 Simulations

We verify our proposed MAB methods through simulations using two methods. First, we run simulations based on data from experiments on real students using a traditional student model. Secondly, we generate random domain graphs and simulate students based on a simple student mastery model.

4.1 Simulations with Real-Student Dataset Using IRT

We use the DBE-KT22 dataset [13] to simulate a domain model and initialize the parameters of the student model. It is a knowledge-tracing dataset based off an undergraduate databases course. The dataset includes a KC-KC mapping which provides the graph structure of the domain and which we utilize to test the efficacy of our algorithms. In addition to this graph structure, the dataset includes over 1000 transaction records of various students who have taken an online practice assessment, containing information regarding the correctness of every student's answer to each question they encountered. To model our simulated learners, we assume every separate transaction instance is representative of an individual student and use their answers to model a sample student in this population. Another piece of information this dataset includes is the ground-truth difficulty of each question, which we use to calibrate the mastery levels of our simulated learners.

We firstly preprocessed the dataset and performed a statistical analysis upon our findings. Upon doing so we analyzed the relative difficulty of mastering each KC i by simply computing the probability of any unique student answering a question involving i correctly, and maintaining a probability distribution for i for every student S. For questions involving multiple KC's i_1, i_2, $\ldots i_n$, answers contribute to each probability distribution equally. Questions are also further subdivided based off the dataset's ground-truth difficulty level. From this we extract, for each KC-difficulty pair, the mean probability of any student from our sample population answering correctly. These distributions are first sampled, then inserted into Eq. (4) to randomly initialize the α values for each of our simulated learner's—such a strategy ensures that our simulation accurately reflects a real-world scenario with realistic student's and mastery levels that are not arbitrarily determined. In addition, we also analyze the correlations between KC correctness, and have found that certain KC's have statistically significant correlations, which also adhere to the KC-KC domain mapping the dataset provides. These correlations were also used in our initialization of simulated learner masteries to represent how students who have mastered one KC may be more likely to master a similar or pre-requisite KC.

Simulations using the DBE-KT22 dataset operated under Item Response Theory (IRT) fundamentals. In IRT, certain models exist to map a student's latent ability in a particular KC to the probability that that student will answer it correctly, such as the 1-parameter logistic model (1PLM). Consider a student s learning within a domain consisting of K knowledge components: each KC $i \in \{1, 2, \ldots, K\}$ is associated with a latent ability θ_i, also referred to as their mastery in i. We utilize an IRT model taken from the Stanford paper "Deep Knowledge Tracing" [14]:

$$P(correct|\alpha, \beta) = c + \frac{1-c}{1+e^{\beta-\alpha}} \tag{4}$$

Where $\alpha = \theta_i$ represents the student mastery, β represents the difficulty to the question given, and c is the probability of guessing a correct answer—in a four-option multiple choice question, this would be 0.25.

The above equation is used to model the behavior of our artificial learners—that is, the student AI used in our simulation. Each simulated learner possesses a particular mastery level α_i for every i, which paired with the above equation allows us to simulate

the correctness of this student's response given any question. These values are ground-truths, initialized upon the creation of the learner, and our model must identify any such α_i that is underneath a given mastery threshold—that is, it is considered weak.

4.2 Simulations with Random Domain Graphs and Simulated Students

In addition to simulating learners using the DBE-KT22 dataset and IRT, we developed a second set of simulations to test the generalizability of our MAB model. In this second set of simulations, we generate random domain graphs for our artificial learners. These random graphs are generated as DAGs using the NetworkX Python library[1]. We confirm that each is acyclic and directed before using them in the simulation. The number of nodes, n_N, in a graph is set to the number of KCs that we would like in our simulated domain model. The number of edges, n_E, is randomly set to a number in the range [2, $n_N \times (n_N - 1)$]. Each student for each simulation is generated with a different random graph. Once the graph is built for a simulated student, the nodes, i.e. the KCs for the student model, must be initialized.

To model the simulated student's ground truth mastery levels, we use the following equation, from [4], to determine their proficiency $\mu_{KC}(s)$:

$$\mu_{KC}(s) = w1 \times apt(s) + w2 \times pre(KC) \tag{5}$$

Where $apt(s)$ is the aptitude of the student and $pre(KC)$ is the ratio of the number of *KC*'s prerequisites the student has mastered to the number of prerequisites *KC* has. The values $w1$ and $w2$ are tunable parameters. The value for $apt(s)$ is set to a random value in the range [0.1, 1] for all simulations except those testing the differences in aptitudes. For those simulations three categories of aptitude are defined: weak, with values in the range [0.1, 0.3]; medium, with values in the range [0.4, 0.6]; and strong, with values in the range [0.7, 1].

The filling in of the student model for a graph is performed in a breadth-first fashion starting with the first layer nodes, i.e. those without any prerequisites. For these nodes $\mu_{KC}(s)$ is set to $apt(s)$, the aptitudes at this layer correspond directly to the student's mastery of the KC. For the second layers and above, the above equation is used to calculate $\mu_{KC}(s)$ taking into consideration the student's mastery of the KC's prerequisites. Because we need to determine both the α and β parameters for each KC, for each node in every layer we generate a random integer value for α in the range [200, 500] and use it along with $\mu_{KC}(s)$ to calculate the β parameter for the KC. In this way we can generate random domain DAGs and translate them into ground-truth student models for testing artificial learners.

When these artificial students provide answers to questions their beta distributions are sampled. This represents answering a question based on the student model we have defined for the artificial learners. If a random value between 0 and 1 is less than the sample, we consider the student to have answered the question correctly. Because we would like to simulate the answering of multiple-choice questions, we have adjusted the sample by adding a guess factor. The guess factor takes into consideration the number of

[1] https://networkx.org/.

choices in a multiple-choice question and represents the probability of a student correctly guessing the answer. The adjusted probability of getting a question correct is defined as:

$$P(correct|\alpha, \beta) = Beta(\alpha, \beta) + \frac{1 - Beta(\alpha, \beta)}{N} \tag{6}$$

Where N is the number of choices in the multiple-choice question. Compare this to the IRT formulation in the previous section. This is a simpler function for the probability that the student will answer a question correctly. It aligns well to our definition of mastery threshold and our Beta-Bernoulli MAB model.

4.3 Experiments and Results

The evaluation of our MAB algorithms is based on the correctness of each algorithm's output weak arm in comparison to the artificial learner's ground-truth weak arms. That is, the algorithm is correct if the output arm $a \in A_W$ where A_W is the set of all ground-truth weak arms as defined by ξ and the artificial learner's KC proficiencies. If $A_W = \varnothing$ then the model is only correct if it has not output an arm. We are also concerned with the number of questions the MAB asks before the algorithm terminates, i.e., the efficiency of the algorithm. Four experiments were conducted for each simulation method using each MAB algorithm.

Experiment 1: KC Count. The number of KCs varies from 2 to 16. The other parameters, error rate, threshold, and aptitude, are kept constant at $\delta = 0.15$, $\xi = 0.7$, student aptitude = medium for DBE-KT22, and $apt(s)$ to a random value between 0.1 and 1 for random graphs, respectively. Results are shown in Figs. 5a and 5b.

Experiment 2: Mastery Threshold. The mastery threshold varies from 0.2 to 0.9 while keeping the other parameters, KC count, error rate, and aptitude, constant. These other parameters are set to KC count = 16, $\delta = 0.15$, student aptitude = medium for DBE-KT22, and $apt(s)$ to a random value between 0.1 and 1 for random graphs, respectively. These results are shown in Figs. 5c and 5d.

Experiment 3: Aptitude. We test three levels of aptitude: weak, medium, and strong. These aptitudes for the random graph experiment are described in Sect. 4.2. For experiments using the IRT model, weak students suffer a flat 0.5 mastery penalty to all KC's, medium students' mastery is not changed, and strong students benefit from a 0.5 increase. The other parameters are set to KC count = 16, $\delta = 0.15$, and $\xi = 0.7$. Results are shown in Figs. 5e and 5f.

Experiment 4: Error Rate. The error rate varies from 0.05 to 0.35 with a step size of 0.05. The other parameters are set to KC count = 16, $\xi = 0.7$, student aptitude = medium for DBE-KT22, and $apt(s)$ to a random value between 0.1 and 1 for random graphs. Results are shown in Figs. 5g and 5h.

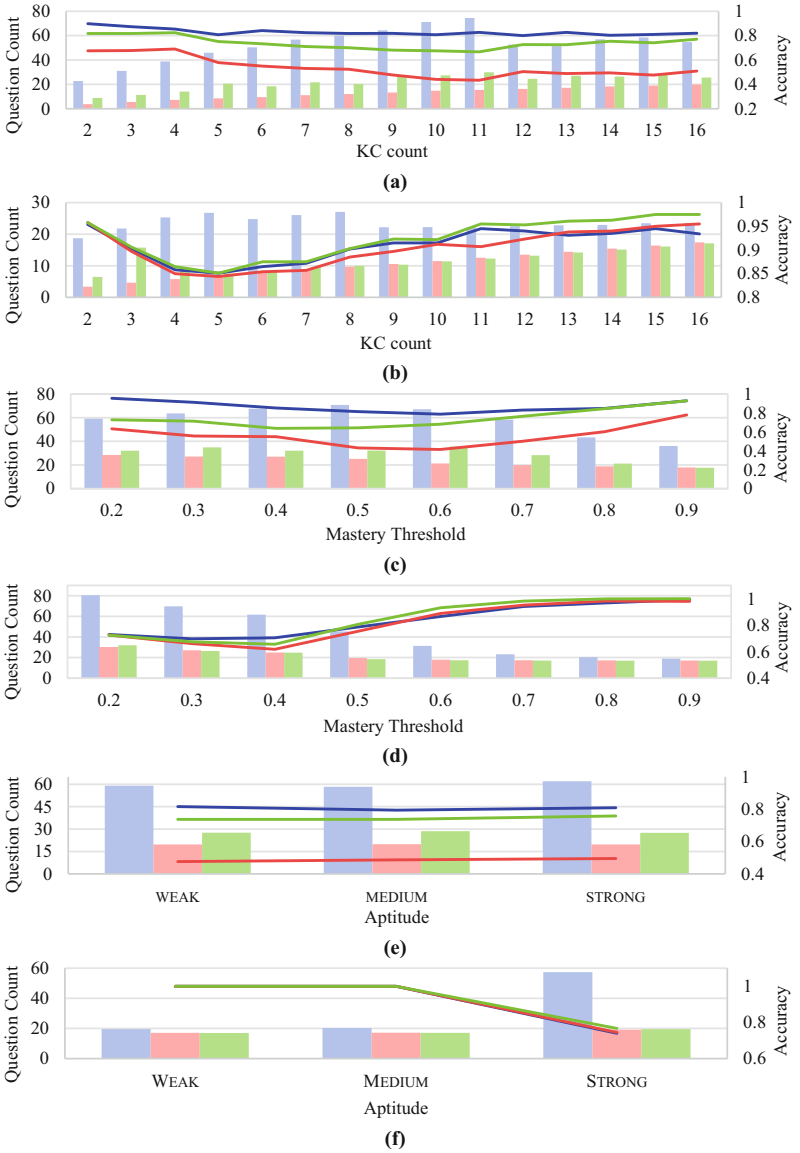

Fig. 5. **(a)** KC count experiments with DBE-KT22 dataset. **(b)** KC count experiments with random graphs. **(c)** Mastery threshold experiments with the DBE-KT22 dataset. **(d)** Mastery threshold experiments with random graphs. **(e)** Aptitude experiments with the DBE-KT22 dataset. **(f)** Aptitude experiments with random graphs. **(g)** Error rate experiments with the DBE-KT22 dataset. **(h)** Error rate experiments with random graphs.

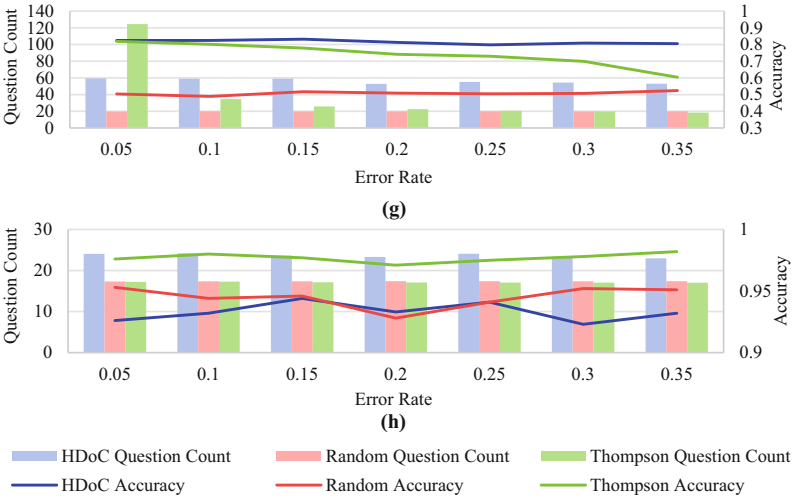

Fig. 5. (*continued*)

4.4 Discussion

Our simulations demonstrate the various tradeoffs between the three MAB algorithms we have developed. In simulations using the DBE-KT22 dataset, we see that Thompson algorithm uses far fewer questions than the HDoC algorithm, whereas HDoC is more accurate. While the random algorithm is the least accurate of the three algorithms in these simulations, it uses the least number of questions. The differences between the algorithm accuracies are less apparent in the second set of simulations using random graphs. In these experiments the Thompson and HDoC algorithms only slightly outperform the random algorithm in each of the experiments. In these experiments, a similar number of questions were asked to the DBE-KT22 experiments. With the HDoC algorithm again asking more questions on average than both Thompson and random algorithms. In general, the data from the two methods of simulation show similar trends. However, in the simulations using random graphs, we see much higher accuracies for each algorithm without as pronounced a difference between each that is seen the DBE-KT22 experiments. The trends that were seen in both simulations show how effective our algorithms are for efficiently finding weak KCs.

From Figs. 5c and 5d, we see how adjusting the mastery threshold ξ changes the accuracy and required questions of the algorithms. Increasing the threshold results in much higher algorithmic accuracy and a lower number of questions required. From Figs. 5e and 5f, we see that our model operates well even with students of varying aptitudes. However, in the random graph simulations, Fig. 5f, we see a sharp drop in accuracy for strong students. This may be due to the differing definitions of mastery and must be examined further. From Figs. 5g and 5h, this experiment provides the most unexpected results. We would expect that with an increased error rate, the number of questions required for our algorithm to return would decrease significantly, but this is not the case. We see that generally, as we increase the allowed error rate for our algorithm a

corresponding decrease in accuracy and sample complexity follows. Note that because here error rate is dependent on the variance of the beta distributions, it is undefined for the naive random selection algorithm. Overall, from the results of our simulations, we believe that the Thompson and HDoC algorithms are good candidates for use in an MAB system for formative assessment. However, there are some limitations and room for improvement. First, we do not consider "partial credit" answers (e.g., based on the usage of hints or on the response time). Also, we did not model the soft or probabilistic pre-requisites among learning objectives in the proficiency model. Second, the proposed algorithm can be easily extended to generate multiple weak skills. Third, by defining the opposite arm selection rule in the algorithm, we can generate the strong learning objectives as adaptive feedback to the student as encouragement.

5 Conclusion and Future Work

We have presented the results of simulations using three different algorithms that can be used in an MAB for identifying weakness or strengths of students from our results we are confident that the Thompson sampling algorithm and, to a lesser extent, the HDoC algorithm would be effective when used in an online adaptive formative assessment system. The main advantage of these algorithms is that they allow the instructor to specify the accuracy and efficiency criteria and that they effectively balance them with a bandit algorithm. From our experiments, we find that the overall performance of the proposed algorithms is quite consistent even after accounting for different variables. The complexity is relatively small and particularly serviceable in a practical setting where questions may be limited in nature and scope. In our future work, we will work to develop methods for automated feedback generation that adapt to varied feedback strategies or students' needs. We will also refine and develop further the random graph model for use in future simulation studies. The differences we discovered between the simulations using the random graph strategy and the simulations using the student dataset will be investigated. Furthermore, we will consider "partial credit" answers (e.g., based on the usage of hints or on the response time) and the soft or probabilistic pre-requisites among learning objectives in the proficiency model. Finally, we will evaluate the proposed approach to measure its actual effectiveness in real-world learning settings.

Acknowledgments. We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC), Alberta Innovates, and Athabasca University, Canada.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Heritage, M.: Formative Assessment: Making it Happen in the Classroom. Corwin, Thousand Oaks (2010)
2. Gareis, C.R.: Reclaiming an important teacher competency: the lost art of formative assessment. J. Pers. Eval. Educ. **20**, 17–20 (2007). https://doi.org/10.1007/s11092-007-9044-5

3. Hattie, J., Timperley, H.: The power of feedback. Rev. Educ. Res. **77**(1), 81–112 (2007). https://doi.org/10.3102/003465430298487
4. Lin, F., De Silva, S.: An approach to generating adaptive feedback for online formative assessment. In: Frasson, C., Mylonas, P., Troussas, C. (eds.) ITS 2023. LNCS, vol. 13891, pp. 88–99. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-32883-1_8
5. Vie, J.J., Popineau, F., Bruillard, É., Bourda, Y.: A Review of recent advances in adaptive assessment. In: Peña-Ayala, A. (eds.) Learning Analytics: Fundaments, Applications, and Trends. Studies in Systems, Decision and Control, vol. 94, pp. 113–142. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-52977-6_4
6. Kano, H., Honda, J., Sakamaki, K., Matsuura, K., Nakamura, A., Sugiyama, M.: Good arm identification via bandit feedback. Mach. Learn. **108**, 721–745 (2019). https://doi.org/10.1007/s10994-019-05784-4
7. Corbett, A.T., Anderson, J.R.: Knowledge tracing: Modeling the acquisition of procedural knowledge. User Model. User-Adap. Inter. **4**, 253–278 (1994). https://doi.org/10.1007/BF01099821
8. Kingsbury, G.G., Houser, R.L.: ICAT: An adaptive testing procedure for the identification of idiosyncratic knowledge patterns. Zeitschrift für Psychologie/J. Psychol. **216**(1), 40–48 (2008). https://doi.org/10.1027/0044-3409.216.1.40
9. Yigit, H.D., Sorrel, M.A., de la Torre, J.: Computerized adaptive testing for cognitively based multiple-choice data. Appl. Psychol. Meas. **43**(5), 388–401 (2019). https://doi.org/10.1177/0146621618798665
10. Lattimore, T., Szepesvári, C.: Bandit Algorithms. Cambridge University Press, Cambridge (2020)
11. Almond, R.G., Mislevy, R.J., Steinberg, L.S., Yan, D., Williamson, D.M.: Bayesian Networks in Educational Assessment. Springer, New York (2015). https://doi.org/10.1007/978-1-4939-2125-6
12. Gupta, S., Chaudhari, S., Joshi, G., Yagan, O.: Multi-armed bandits with correlated arms. IEEE Trans. Inf. Theory **67**(10), 6711–6732 (2021). https://doi.org/10.1109/TIT.2021.3081508
13. Abdelrahman, G., Abdelfattah, S., Wang, Q., Lin, Y.: DBE-KT22: a knowledge tracing dataset based on online student evaluation (2022). https://doi.org/10.21203/rs.3.rs-2192747/v1
14. Piech, C., et al.: Deep knowledge tracing. In: Advances in Neural Information Processing Systems (2015). https://doi.org/10.48550/arXiv.1506.05908