CurriculumTutor: a novel tutoring algorithm for mastering a curriculum using adaptive activity sequencing

Shabana K M[1], Chandrashekar Lakshminarayanan[2], and Jude K Anil[3]

[1] Indian Institute of Technology Palakkad shabana.meethian@gmail.com
[2] Indian Institute of Technology Madras chandrashekar@cse.iitm.ac.in
[3] Indian Institute of Technology Palakkad judekanil@gmail.com

Abstract. An important problem in an intelligent tutoring system (ITS) is that of adaptive sequencing of learning activities in a personalised manner so as to improve learning gains. In this paper, we consider intelligent tutoring in the *learning by doing* (LbD) setting, wherein the *concepts* to be learnt along with their inter-dependencies are available as a *curriculum graph*, and a given concept is learnt by performing an activity related to that concept (such as solving/answering a problem/question). For this setting, recent works have proposed algorithms based on multi-armed bandits (MAB), where activities are adaptively sequenced using the student response to those activities as a direct feedback. In this paper, we propose CurriculumTutor, a novel technique that combines a MAB algorithm and a *change point detection* algorithm for the problem of adaptive activity sequencing. Our algorithm improves upon prior MAB algorithms for the LbD setting by (i) providing better learning gains, and (ii) reducing hyper-parameters thereby improving personalisation. We show that our tutoring algorithm significantly outperforms prior approaches in the benchmark domain of two operand addition up to a maximum of four digits.

Keywords: adaptive sequencing · multi-armed bandits · change-point detection · personalisation

1 Introduction

Personalised learning approaches tailored to address the individual needs, skills, and interests of each student, have been found to cause a significant improvement in learning gains for the students, apart from providing an engaging learning experience. Intelligent Tutoring Systems (ITS) have been effective in delivering personalised learning to students in an automatic manner. An ITS consists of three important components namely: (i) *domain model* that captures the relations or dependencies between the various concepts to be learnt, (ii) *student model* that represents the student's current knowledge level and how it changes as the tutoring progresses and (iii) *tutoring model* that decides the sequence of learning activities presented to students. The tutoring model performs *adaptive*

sequencing of activities, wherein learning activities to be presented to students are selected based on student competence levels estimated through their interactions with the system. Such adaptive sequencing leads to better personalisation and hence faster learning curves.

Clement et. al. [4] proposed a tutoring algorithm called ZPDES that uses a *curriculum graph* capturing the concept inter-dependencies as a domain model. The key highlight of ZPDES is the use of ideas from multi-armed bandits to perform adaptive assignment of learning activities based on student responses. Later Brunskill et. al. [10], used ZPDES to propose an adaptive sequence of activities to advance students through a curriculum graph generated based on an algorithmic representation of the concepts for the domain of two operand addition upto a maximum of 4 digits.

Our Contribution: In this paper, we consider the *learning by doing* (LbD) setting used in [10], wherein the concepts to be mastered are available in the form of a curriculum graph and the students learn a concept by performing activities related to the concept. Even though ZPDES is less reliant on the underlying student learning model, it has its own set of hyper-parameters which adversely affect personalisation. Our main contribution is a novel tutoring algorithm called CurriculumTutor for the LbD setting. CurriculumTutor combines the ideas of multi-armed bandits and *change point detection*. Here, the change point detector separately tracks the mastery level of each activity and an upper confidence based MAB algorithm manages the exploration vs exploitation tradeoff. We show that our CurriculumTutor significantly outperforms ZPDES in the benchmark domain of two operand addition up to a maximum of four digits, especially in scenarios where the learner faces difficulty in learning a subset of concepts in the curriculum graph.

2 Background

In this section, we first describe the learning by doing setting and then explain how prior works have used ideas from multi-armed bandits in the tutoring model.

Learning by Doing (Fig. 2): A *concept* is a representational unit of educational content. A concept is learnt by performing an activity related to that concept (such as solving/answering a problem/question). A *concept/curriculum graph* has concepts as nodes and the edges represent dependency/pre-requisite relations between concepts. Several methods have been proposed in the literature to construct concept graphs from text corpus [6][13]. In this paper, we assume access to a concept graph for the curriculum to be mastered. *Zone of Proximal Development (ZPD)* is a set of concepts on the boundary of the student's knowledge (Fig. 1). It is based on an idea from classical psychology and education research which states that learning is the fastest and most engaging when practicing on material slightly beyond the current abilities of the student

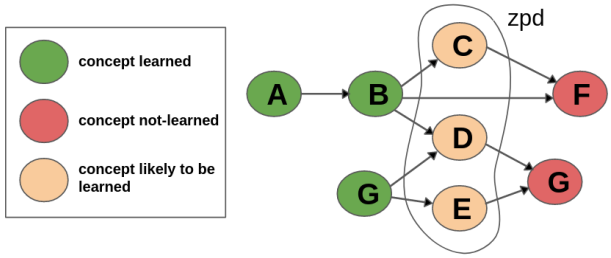


Fig. 1: Curriculum graph and ZPD

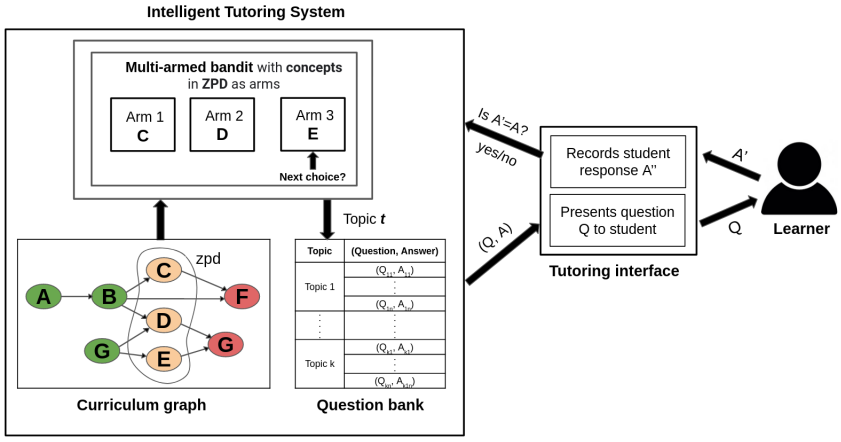


Fig. 2: Multi-armed bandit framework for adaptive activity sequencing in an ITS

[3]. A *Question bank* contains a collection of question answer pairs associated with each concept. It is further assumed that all the questions associated with a concept are of the same difficulty level.

Multi-armed bandit (MAB) problem consists of a fixed set of actions (arms), and an unknown reward distribution associated with each arm. The system has to select actions (arms) such that its long term cumulative reward is maximized. The key challenge in this problem is to balance exploiting the information that has already been gained about the effectiveness of each action and exploring actions where the estimates about their value are still relatively uncertain, referred to as the *exploration-exploitation* tradeoff. Over time the system learns which actions are more effective and can earn larger rewards.

MAB applied to Tutoring Model: In the multi-armed bandit framework for adaptive sequencing, the activities/lessons in the ZPD form the arms and rewards are computed based on student responses, as seen in Fig. 2. The system

Algorithm 1 ZPDES

Initialization: For each activity a in ZPD in the `not-learned` state set $w_a = w_0$

Selecting the next activity: The weights (w_a) of the activities in the ZPD are normalized ($w_{a,n}$) and an activity is sampled from the probability distribution p_a

$$w_{a,n} = \frac{w_a}{\sum_{a' \in ZPD} w_{a'}}, \;\; p_a = w_{a.n}(1-\gamma) + \gamma \frac{1}{|ZPD|} \;\; a \in ZPD$$

Reward for the chosen activity a based on correctness $C_{a,i}$:

$$r_{a,i} = \sum_{k=n_a-d/2}^{n_a} \frac{C_{a,k}}{d/2} - \sum_{k=n_a-d}^{n_a-d/2+1} \frac{C_{a,k}}{d/2}$$

where n_a is the number of times activity a has been chosen

Updating weight of activity a: $w_a \leftarrow \beta w_a + \eta r_{a,i}$

maintains an estimate of the learning progress associated with each concept in the ZPD, based on which it decides the next activity to be offered to the student. In the process of maximizing cumulative rewards, the MAB framework would pick activities from the concepts that the student tends to learn faster, thereby generating a personalized activity sequence. This formulation is less reliant on the underlying model of student learning and can therefore better adapt to different individual learning behaviors.

3 Prior Work and Research Gaps

In this section, we discuss the ZPDES algorithm proposed by Clement et. al. [4] and then look at the research gaps in ZPDES, in particular about how the hyper-parameters in ZPDES adversely affects personalization.

Zone of Proximal Development and Empirical Success (ZPDES) [4] (see Algorithm 1) is based on the multi-armed bandits framework for adaptive activity sequencing where pre-conditions between activities are provided in the form of an expert defined curriculum graph. At each time-step, ZPDES selects the next activity based on the normalized weights of activities ($w_{a,n}$) in the ZPD as well as an exploration factor γ(Algorithm 1). The reward at time t is computed by taking the difference of the number of successes in the last $d/2$ samples with the $d/2$ previous samples, where d is the window-size. The reward provides an empirical measure of how the success rate is increasing. The reward becomes close to zero when either a concept has been mastered or the student struggles to master it. These rewards are then used to update the weight parameters (w_a). However ZPDES doesn't propose a method to determine if an activity has been mastered or not.

Later Brunskill et. al. [10] proposed a novel system that combines automatic curriculum ordering [1] with ZPDES for automatically and adaptively advancing a student through a curriculum. A 'sliding windowed' average was used in this framework for inferring mastery, where a topic is determined to be *mastered* when the accuracy over the past d attempts reaches above a specified threshold t.

Research Gap: The major drawback of ZPDES is the use of many hyper-parameters such as d, w_0, γ, β and η, which are critical for its deployment, and eventual success. Using fixed values for these hyper-parameters would have an adverse effect on personalisation since the same set of hyper-parameters may not be effective for the entire population. On the contrary, tuning these hyper-parameters for different individuals is an additional computational overhead and a time consuming process.

4 Our Approach: Breaking Adaptive Activity Sequencing into Sub-Problems

Our algorithm `CurriculumTutor` addresses the research gap in the previous section. We describe our approach in this section and present the algorithm in the next section. The main reason for the presence of hyper-parameters in ZPDES is that it tries to address together both the exploration vs exploitation tradeoff as well as the non-stationarity of the student responses. On the contrary, our approach is to decompose the adaptive activity sequencing problem into the following two sub-problems namely (i) **activity selection** to take care of the exploration vs exploitation tradeoff, and (ii) **detection of mastery** to take care of the non-stationarity of the student responses. We now describe these in detail.

4.1 Activity selection

The tutoring algorithm at each time step has to select an activity that offers higher learning progress to the student. In the multi-armed bandit framework for activity selection, the concepts in the ZPD form the arms while rewards are computed based on student responses. We consider the case where the rewards are binary, i.e., a reward of one is obtained if an activity related to the selected concept is correctly performed and otherwise zero. The problem can now be modelled as a *Bernoulli multi-armed bandit*, where a reward of 1 is obtained with probability p that is dependent on the knowledge level of the associated concept(arm).

4.2 Detection of mastery

The tutoring algorithm has to infer whether a concept has been mastered based on the sequence of student responses. Determining whether a concept has been mastered helps to avoid over-practicing which leads to a drop in the learning gains, and under-practicing that results in the concept not being learned.

Each concept is assumed to be associated with two states - *learned* and *not-learned*. The probability of getting a correct response in the non-learned state is given by p_g, the guess probability. The probability of answering incorrectly while in the learned state is given by the slip probability, p_s. Thus the i^{th} response r_i is distributed as follows:

$$r_i \sim \begin{cases} Bernoulli(p_g) & \text{when } state_i = \text{not-learned} \\ Bernoulli(1-p_s) & \text{when } state_i = \text{learned} \end{cases}$$

The reward distribution is now *piecewise stationary* i.e., remains constant for a certain period, and shifts at some unknown time step referred to as a change-point, that corresponds to the transition from the *not-learned* to *learned* state. Detection of mastery of a concept can now be modelled as a *change-point detection problem*, which is formally described as follows [9]: Suppose $X_1, X_2, ...$ are independent random variables observed sequentially and $X_1, ..., X_{m-1}$ have distribution function F_0 for $m \in \{1, 2, ...\}$ while $X_m, X_{m+1}, ...$ have distribution function $F_1 \neq F_0$.

A sequential change-point detection procedure is identified with a stopping time τ for an observed sequence $\{X_n\}_{n\geq 1}$. The *average detection delay (ADD)* and *FAR* of a detection procedure is defined as follows [12]:

$$ADD_m(\tau) = E_m(\tau - m | \tau \geq m)$$
$$FAR(\tau) = \frac{1}{E_0(\tau)}$$

where $E_0(\tau)$ denotes the expectation of the sequence $\{X_n\}_{n\geq 1}$ when there is no change, i.e., $m = \infty$. A good detection procedure should have a low FAR and small values of the expected detection delay.

5 Our Algorithm: CurriculumTutor

We now describe CurriculumTutor, a novel tutoring algorithm that combines a Bernoulli multi-armed bandit algorithm with a change point detection technique for adaptive activity sequencing.

CurriculumTutor uses the *Kullback-Leibler Upper Confidence Bound(KL-UCB)* algorithm to perform activity selection, as it is shown to have better performance bounds than UCB and its variants [8]. The change-point detection algorithm used by CurriculumTutor is *Cumulative Sum (CUSUM)*. Here the log-likelihood ratio (LLR) is used to test the hypotheses that a change occurred at the point λ and that there is no change at all ($\lambda = \infty$) which is defined as

$$Z_{n,\lambda} = \sum_{k=\lambda}^{n} log \frac{p_1(X_k | X_1, ..., X_{k-1})}{p_0(X_k | X_1, ..., X_{k-1})}, \;\; n \geq \lambda$$

Algorithm 2 CurriculumTutor

Initialize zpd = Minimal elements of the partial ordering curriculum, P
Add arms corresponding to elements in zpd
$t = 1$
repeat
  c_t =Concept selected by KL-UCB
  Present an activity of type c_t and observe correctness of response r_t
  Update estimates associated with c_t based on r_t
  Update CUSUM estimate for c_t
  if change-point detected for c_t **then**
    Mark c_t as *learned*
    Remove c_t from zpd
    $zpd \leftarrow zpd \cup \{q' \in P \mid (q_t \leq q') \wedge (\nexists\, p \in P \text{ such that } p \leq q' \wedge p \text{ is not-learned})\}$
    Add arms corresponding to new elements added in zpd
  end if
  $t \leftarrow t + 1$
until zpd is empty

Here, p_0 and p_1 are the pre-change and post-change probability density functions respectively. In CUSUM, the maximum LLR statistic $U_n = max_{1 \leq \lambda \leq n} Z_{n,\lambda}$ is compared with a threshold h and a change is detected when the value of U_n exceeds h. When $h > 0$ and the observations are independent and identically distributed (i.i.d.), U_n can be replaced by the statistic $\tilde{U}_n$ which obeys the recursion:

$$\tilde{U}_n = max \left\{ 0,\ \tilde{U}_{n-1} + log\, \frac{p_1(X_n)}{p_0(X_n)} \right\}$$

with the initial condition $\tilde{U}_0 = 0$. It has been proved [9] that in the i.i.d. case CUSUM minimizes the worst case average detection delay among all the detection algorithms for which the FAR is fixed at a given level $\overline{FAR}$.

The guess and slip probabilities, p_g and p_s for each concept can be defined by the user. The pre and post change distributions can then be defined as follows:

$$p_0(k) = {p_g}^k (1 - p_g)^{(1-k)}$$
$$p_1(k) = (1 - p_s)^k (p_s)^{(1-k)} \quad \text{for } k \in \{0, 1\}$$

The $\tilde{U}_n$ statistic is used for change-point detection, as the correctness of student responses are independent random variables given the state of each skill.

A concept is removed from the ZPD when it is *determined* to have been transitioned into the learned state. The ZPD is then updated with concepts in the not-learned state all of whose prerequisites are in the learned state. This continues until the student has mastered all the concepts in the curriculum graph. The *CurriculumTutor* algorithm has been described in Algorithm 2.

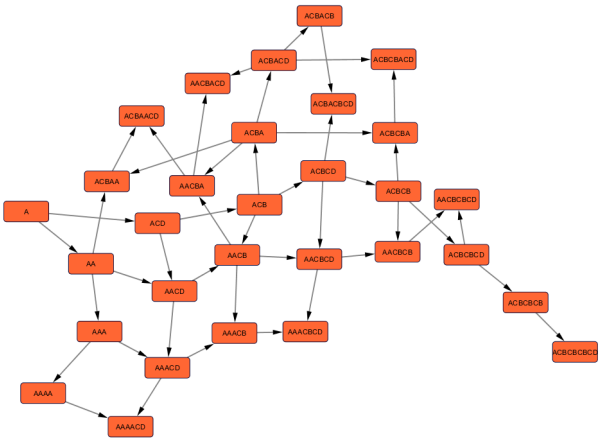


Fig. 3: Curriculum graph for 2 operand addition upto maximum of 4 digits

6 Experiment Setup

We will now describe the experimental setup to evaluate our algorithm in the domain of two operand addition with simulated students. The curriculum graph generation for the domain and models for simulated students are described below. While the curriculum graph is used by our algorithm, the simulated students are used only to evaluate our algorithm, and not by the algorithm itself.

6.1 Generating the curriculum graph

The trace based framework proposed by Andersen et al. [1] was used to generate a curriculum graph for the domain of two operand addition up to a maximum of four digits. Four basic operations were identified to be required to solve an integer addition problem: one digit addition without a carry (A), one-digit addition with a carry (B), writing a carry (C), and bringing down a final carry (D). For instance, problems can be decomposed into basic skills as shown in Table 1:

These traces can then be ordered by complexity based on the N-gram-based partial ordering as follows: *Let n be any positive integer. A trace $T1$ is said to be at least as complex as trace $T2$ if every n-gram of trace $T2$ is also present in trace $T1$*. The curriculum graph generated with n=3 has been given in Fig 3.

Problem	2+3	15+18	93+15	298+865
Trace	A	ACB	AACD	ACBCBCD

Table 1: Traces generated for addition problems

6.2 Student Models for Simulation

Students were simulated using the following popular knowledge tracing models. For each student model, separate knowledge components were defined corresponding to addition of different lengths, presence of carry and overflow.

Bayesian Knowledge Tracing (BKT) BKT [5] models a student's knowledge state as a two-state Hidden Markov Model(HMM), one per knowledge component/skill, where the skill is either mastered by the student or not. Prerequisites between knowledge components were enforced by varying $p(T)$ depending on whether the prerequisite was learned or not.

Learning Factors Analysis (LFA) In LFA [2], the probability p of a student performing correctly on an activity is modeled as a logistic function with parameters: n_i - number of times knowledge component i has been practiced, α - student learning parameter, β_i - coefficient of knowledge component i, γ - difficulty level of the activity

$$ln(\frac{p}{1-p}) = \frac{\alpha}{|P_{kc}|} \sum_{i \in P_{kc}} n_i \beta_i$$

where kc is the knowledge component associated with the activity
P_{kc} is the set containing kc along with its pre-requisite knowledge components

Performance Factor Analysis (PFA) In PFA [11], the probability p of a student performing correctly on an activity is modeled as a logistic function with parameters: s_i - number of times knowledge component i has been used correctly, f_i - number of times knowledge component i has been used incorrectly, α - student learning parameter, β_i - coefficient for the success count of knowledge component i, η_i - coefficient for the failure count of knowledge component i, γ difficulty level of the activity

$$ln(\frac{p}{1-p}) = \frac{\alpha}{|P_{kc}|} \sum_{i \in P_{kc}} (s_i \beta_i + f_i \eta_i)$$

Integrating Knowledge Tracing ans Item Response Theory (KT-IRT) KT-IRT [7] combines the BKT model with item response theory. Here the student's knowledge is represented as a HMM with binary states. The probability of performing correctly on an activity at the i^{th} attempt is given by:

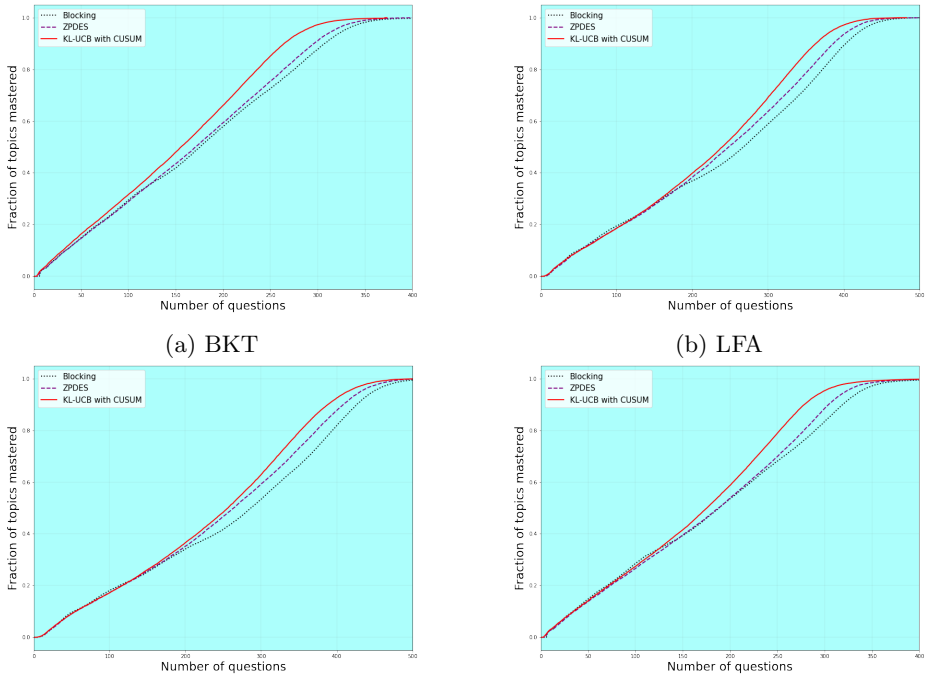
$$p_i = \begin{cases} p_g(\frac{1}{1+e^{-(\beta_k n_k + c_{k,0})}}) & \text{when } state_i = \text{not-learned} \\ (1-p_s)(\frac{1}{1+e^{-(\beta_k n_k + c_{k,1})}}) & \text{when } state_i = \text{learned} \end{cases}$$

6.3 Parameters

Hyper-parameters for ZPDES: The best value of hyper-parameters such as η, β, γ and w_0 were found using grid search separately for each student model.

Change-point detection: For selecting the window size d and accuracy threshold t for ZPDES, and the threshold h for CUSUM, the FAR was fixed as 5e-05. To find the FAR of sliding window based method, a markov chain was constructed with the states corresponding to the number of ones to be obtained to reach the accuracy threshold d. With zero set as the absorbing state, the mean time to absorption for the chain was computed, which corresponds to $E_0(\tau)$. Similarly for CUSUM, a markov chain was constructed with the states corresponding to the possible values taken by the statistic $\tilde{U}_n$ and states with values greater than h were designated as the absorbing states. Again, the mean time to absorption was calculated to compute the FAR.

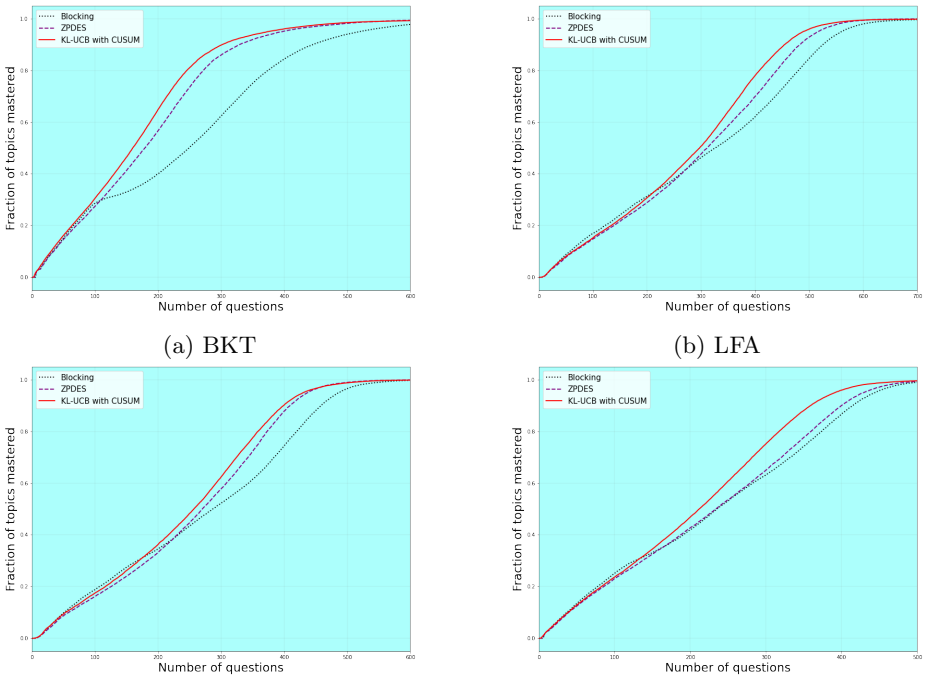
For the sliding window, the window size was set as 8 and the accuracy threshold as 0.7. Thus, out of the last 8 activities the student has to get at least 6 of them correct in order to inferred that the concept has been mastered. Similarly



(a) BKT (b) LFA

(c) PFA (d) KT-IRT

Fig. 4: Scenario 1 - Student not struggling with any of the concepts



(a) BKT (b) LFA

(c) PFA (d) KT-IRT

Fig. 5: Scenario 2 - Student struggling with a few concepts in the intermediate phase

the threshold h for CUSUM was set as $ln(1/0.0004)$. For CUSUM, the pre and post change probabilities were sampled from $Beta(20, 40)$ for each concept.

7 Results

We will now present experimental results that compare the performance of three algorithms namely CurriculumTutor, ZPDES and blocking (a basic scheduling scheme where activities associated with the same concept are presented to a student until the concept is mastered uses a sliding window to infer mastery). For these three algorithms, the following two scenarios were simulated using the four different student models:

Scenario 1 Student not struggling with any of the concepts
Scenario 2 Student struggling with a few concepts in the intermediate phase

Learning gains were measured by computing the fraction of concepts in the curriculum graph mastered by the student over a given number of questions. The average curves for 500 runs were plotted, as seen in figures 4 and 5. We observed that **CurriculumTutor performs better than ZPDES and Blocking** in both the scenarios, irrespective of the student models chosen.

8 Conclusion and Future Work

In this paper, we have proposed *CurriculumTutor*, a novel multi-armed bandit and change-point detection based algorithm for adaptive activity sequencing to master a curriculum in the learning by doing setting. Simulation results show that CurriculumTutor significantly out-performs prior approaches in the benchmark domain of two operand addition up to a maximum of four digits. Future work on this problem would include accounting for forgetting of a learned concept, considering partial credit scores and accommodating for activities with varying levels of difficulty associated with a concept.

References

1. Andersen, E., Gulwani, S., Popovic, Z.: A trace-based framework for analyzing and synthesizing educational progressions. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. pp. 773–782 (2013)
2. Cen, H., Koedinger, K., Junker, B.: Learning factors analysis–a general method for cognitive model evaluation and improvement. In: International Conference on Intelligent Tutoring Systems. pp. 164–175. Springer (2006)
3. Chaiklin, S.: The zone of proximal development in vygotsky's analysis of learning and instruction. vygotsky's educational theory in cultural context, 1, 39-64 (2003)
4. Clement, B., Roy, D., Oudeyer, P.Y., Lopes, M.: Multi-armed bandits for intelligent tutoring systems. arXiv preprint arXiv:1310.3174 (2013)
5. 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)
6. Gordon, J., Zhu, L., Galstyan, A., Natarajan, P., Burns, G.: Modeling concept dependencies in a scientific corpus. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 866–875 (2016)
7. Khajah, M.M., Huang, Y., González-Brenes, J.P., Mozer, M.C., Brusilovsky, P.: Integrating knowledge tracing and item response theory: A tale of two frameworks. In: CEUR Workshop proceedings. vol. 1181, pp. 7–15. University of Pittsburgh (2014)
8. Lattimore, T., Szepesvári, C.: Bandit algorithms. Cambridge University Press
9. Lorden, G., et al.: Procedures for reacting to a change in distribution. The Annals of Mathematical Statistics **42**(6), 1897–1908 (1971)
10. Mu, T., Goel, K., Brunskill, E.: Program2tutor: Combining automatic curriculum generation with multi-armed bandits for intelligent tutoring systems. In: Conference on Neural Information Processing Systems (2017)
11. Pavlik Jr, P.I., Cen, H., Koedinger, K.R.: Performance factors analysis–a new alternative to knowledge tracing. Online Submission (2009)
12. Tartakovsky, A.G., Rozovskii, B.L., Blazek, R.B., Kim, H.: A novel approach to detection of intrusions in computer networks via adaptive sequential and batch-sequential change-point detection methods. IEEE transactions on signal processing **54**(9), 3372–3382 (2006)
13. Wang, S., Ororbia, A., Wu, Z., Williams, K., Liang, C., Pursel, B., Giles, C.L.: Using prerequisites to extract concept maps fromtextbooks. In: Proceedings of the 25th acm international on conference on information and knowledge management. pp. 317–326 (2016)