Program2Tutor: Combining Automatic Curriculum Generation with Multi-Armed Bandits for Intelligent Tutoring Systems

Tong Mu
Stanford University
tongm@stanford.edu

Karan Goel
Carnegie Mellon University
kgoel93@gmail.com

Emma Brunskill
Stanford University
ebrun@cs.stanford.edu

Abstract

Tutoring systems can present material in a curriculum to help students reach mastery on a set of related topics. For a domain, determining how the topics are related and designing accurate statistical models of student learning is a complex process that often involves significant time and expertise. Automating more of this process could extend the reach and impact of intelligent tutoring systems. Prior work has shown how a prerequisite structure or knowledge graph of the related material can be automatically constructed given an algorithmic representation of the underlying material. Other work has shown how to automatically progress a student through a knowledge graph with minimal assumptions about a model of the underlying student learning process. We present preliminary work on uniting these two ideas in the domain of addition. We also include a novel approach for identifying the initial background knowledge of the learner. Our early results using simulated students suggest that this approach better tailors the number and kinds of problems proposed to students in order to master all associated topics compared to non-adaptive, hand designed curricula.

1 Introduction and Related Work

Intelligent tutoring systems (ITS) can make education more effective by individualizing teaching to each student. This can mitigate some of the problems that arise when presenting all students with the same learning schedule, such as receiving less or significantly more practice activities than needed [7]. Learning in these systems can be optimized by building on ideas from classical psychology and education research which states that learning is fastest and most engaging when practicing on material slightly beyond the current abilities of the student, activities in the Zone of Proximal Development (ZPD)[3]. To realize this, it is necessary to have an effective ordering of activities into a curriculum, a method of assessing the initial knowledge state of the student, and a method for proposing activities to students to advance them through the curriculum to optimize learning.

Automating this pipeline can make ITSs more scalable and impactful. In this work we unify ideas of automatic curriculum generation from execution traces [1, 9] and automatic problem selection using reinforcement learning techniques [5, 8]. Specifically we use the multi-armed bandit algorithm for problem selection (ZPDES) of Clement et. al [5] as it is less reliant than other methods on the underlying student learning model which can be advantageous. This allows it to generalize better to student models with different parameters [4] in simulation. Additionally we build on prior work in probabilistically detecting the knowledge boundary of the student [6, 9] and present a novel approach to determine the initial knowledge state of the student within the curriculum.

We assess our unified approach by simulating students modeled using the Bayesian Knowledge Tracing (BKT) model [2] and show this approach can adapt to different students, presenting them

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

Given Trace	Action Description
A	Move summand 0 pointer left.
B	Add summand 1 to column sum
C	Move summand 1 pointer left.
D	Add summand 2 to column sum
E	Move carry pointer left.
F	Add carry to column sum
G	Write carry
H	Move output pointer left.
I	Add final overflow to column sum
J	Output writes out.

Figure 1: Action labels and descriptions for two operand addition.

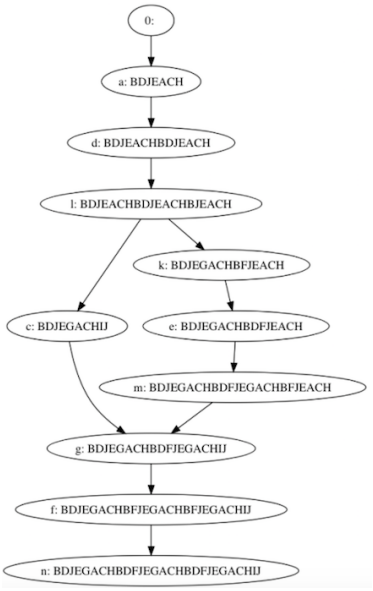

Figure 2: Progression ordering over 10 problems.

with different types and frequencies of problems that would allow them to learn faster than a fixed, hand designed progression.

2 Automatic Curriculum Generation from Execution traces

In this section, we present a summary from the work of Andersen et al. [1] on synthesizing progressions from execution traces which we apply in our idea to the domain of two operand addition.

Given a program for solving an addition problem, we can label the actions taken to solve the problem to build an execution trace of the solution. We can then order these traces by complexity using the following notion from [1]:

Definition 2.1 (Andersen et al. [1])**.** Let n be any positive integer. We say that a trace T_1 is at least as complex as trace T_2 if every n-gram of trace T_2 is also present in trace T_1.

An example of action labels and an ordering over 10 problems using this notion of difficulty is shown in Figure 1 and Figure 2. To create a full progression, problems were randomly generated and grouped by trace. While generating problems in this manner is sufficient for this addition domain, it is an area of future work on automatically generating these problems in other domains. We consider two operand addition problems with operands between one and four digits and our final curriculum is represented by an ordered graph over 49 different traces.

3 Progressing Students Using Multi-Armed Bandits

In this section, we summarize the algorithm proposed by Clement et al. [5] for using multi-armed bandits for problem selection which we use in our idea. This algorithm at each timestep selects a problem from within the set of problem types on the boundary of the student's knowledge, or the ZPD, that it predicts will give the most reward, which is measured in terms of student learning progress. A brief summary of the algorithm is described.

On initialization, all problems start with an initial unnormalized weight $w_a = w_i$. The weights of the problems (w_a) in the ZPD are normalized (w_a^n) to ensure a proper probability distribution (p_a) with which a problem from the ZPD is selected:

$$w_a^n = \frac{w_a}{\sum_{a' \in ZPD} w'_a} \qquad p_a = w_a^n(1-\gamma) + \gamma\frac{1}{|ZPD|} \quad a \in ZPD \tag{1}$$

Where γ is the hyperparameter for the rate of exploration and $|ZPD|$ is the number of elements in the ZPD.

Once a problem is selected, it is presented to the student and correctness of the student answer for the i^{th} time the problem is presented is recorded as C_i. The reward of the problem is calculated by approximated gradient of the performance of the student on that problem and this reward is used to update the weight of the problem to account for the non-stationarity of the rewards:

$$r_a = \sum_{k=t-d/2}^{t} \frac{C_k}{d/2} - \sum_{k=t-d}^{t-d/2} \frac{C_k}{d-d/2} \qquad w_a \leftarrow \beta w_a + \eta r_a \tag{2}$$

Where d is the hyperparameter for the length of history to be considered and β and η are hyperparameters for update rates.

A problem type is removed from the ZPD and the ZPD is progressed after student accuracy on that problem type over d reaches an accuracy of threshold hyperparameter h.

4 Finding the initial ZPD

We present a probabilistic greedy, entropy based algorithm for calculating the initial ZPD for initializing the ZPDES algorithm. We make two assumptions, firstly, students that have mastered all concepts relevant to a problem i are assumed to have also mastered all concepts relevant to problems at most as difficult as i defined by the curriculum. Secondly, the students' answers will be noisy; there is both a chance they will guess the answer correctly without mastering the concepts and a chance they will slip and answer incorrectly while having mastered the concept. We build on the graph coloring algorithm from [9] to create an entropy based graph coloring algorithm that works in the presence of this noise.

On initialization all nodes are uncolored. Each node keeps track of a correct count: c and incorrect count: ic initialized to 0. At each time step, the algorithm will propose a problem and receive an answer from a student. The correctness of the answer is recorded and updated for the problem and all relevant dependencies. For example, if the student is able to solve problem s, the c for all problems s' that are at most as hard as s is incremented by one. If the student is queried again about this problem and the answer differs from the previous answer to this problem, the previous count update is reverted. At each point, the entropy of the problem is calculated and used as the weight of that node:

$$w_i = entropy(\frac{c_i + c_0}{ic_i + c_0}) \tag{3}$$

where c_0 is a regularizing factor. Let n_i^+ represent the set of all uncolored problems at most as difficult as problem i and n_i^- represent the set of uncolored problems at least as difficult as problem i. Then we can define the total weights tw_i^+ and tw_i^- of a node i in the uncolored set and greedily chose the next problem, i^* to present to students:

$$tw_i^+ = \sum_{j \in n_i^+} w_j \quad \text{and} \quad tw_i^- = \sum_{j \in n_i^-} w_j \qquad i^* = argmax \min((tw_i^+, tw_i^-)) \quad i \in uncolored \tag{4}$$

We color a node an all relevant dependencies once the entropy of the node decreases below a threshold. An area of future work is to theoretically analyze and improve this algorithm.

5 Simulations and Simulation Results

We simulated students using the BKT model which is popularly used in the education literature [2]. BKT models a student's knowledge state of a knowledge component as a two-state Hidden Markov Model. The model transitions from the not-learned to mastery with probability $p(T)$, and once mastered a knowledge component cannot be forgotten. In the not-learned state, a student can guess the solution of a problem correctly with probability $p(G) < 0.5$. In the mastered state, a student can slip and answer incorrectly with probability $p(S) < 0.5$.

For our student model, each problem is decomposed into knowledge components corresponding to addition of different lengths, carrying, and handling overflow. We enforce prerequisites between knowledge components by varying $p(T)$ depending on whether the prerequisite was learned or not.

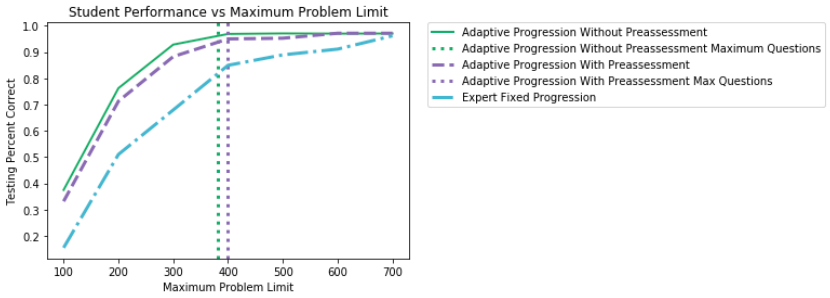

Figure 3: Test performance vs Problem Limit for Student 3.

For example $p(T)$ for the knowledge component of two-digit plus two-digit addition is $p(T_{\text{high}})$ if the prerequisite of one digit plus one digit addition is learned, and is $p(T_{\text{low}})$ if the prerequisite is not learned. To sample the answer of the student on a given problem, we compute the overall probability of the student answering correctly by multiplying the probability of correctness of the student on each knowledge component relevant to the problem.

We simulated students with different parameters $p(T)$, $p(S)$ and $p(G)$ as well as different prerequisite structures between knowledge components. We compare our idea of automatically generating a curriculum and progressing students to a baseline of a fixed progression modeled off of progressions found in addition worksheet from various education websites online that successively presents students with n problems per problem type over 10 different problem types grouped and sorted by the length of the operands of the problem. The effectiveness of each system was evaluated by finding the percent of problems answered correctly by each student after on a fixed test set of 1000 randomly generated problems. All results are the averaged over 100 different trials of each Student-Progression pair.

From our initial results, we found that our system with fixed parameters is able to adapt well to all the students compared to the fixed progression where drastically different values of n were needed for different students. Additionally, while preassessment slightly increased the number of problems needed for student models with all initial knowledge components in the not learned state (Student1 and Student2), it drastically decreased the number of problems needed for student models with knowledge components in the known state (Student3). These result are shown in Table 1. Additionally we ran our system and capped the number of problems presented to students to mimic a time constraint. As shown in Figure 3 for one of the students (Student3) we find that our system with and without pre-assessment can lead to more efficient learning in constrained time than a fixed progression.

Table 1: Problems needed to master all knowledge components

Student Model	ZPDES without Pre-assessment	ZPDES with Pre-assessment	Fixed Progression and n
Student1	262	267	300 (n = 30)
Student2	382	399	800 (n = 80)
Student3	216	13	100 (n = 10)

6 Conclusion and Future Work

We proposed a novel system for automatically creating an intelligent tutoring system starting from execution traces by unifying previous work on automatic curriculum generation and automatic problem selection. We introduce a probabilistic entropy based pre-assessment algorithm that drastically reduces the number of problems necessary for students that have mastery of the material and we compare our system to a fixed baseline and show that our system can better adapt to students and progress them faster.

We plan to investigate several areas of future work including (i) automatically generate practice problems so that they induce coverage over the space of possible execution traces; (ii) a theoretical analysis of our pre-assessment algorithm; (iii) leveraging richer reward signals such as the difference in the amount of time spent by the student on distinct problems; (iv) extending to domains beyond two operand addition; (v) running a real-world experiment.

References

[1] E. Andersen, S. Gulwani, and Z. Popovic. A trace-based framework for analyzing and synthesizing educational progressions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 773–782. ACM, 2013.

[2] J. R. Anderson, A. T. Corbett, K. R. Koedinger, and R. Pelletier. Cognitive tutors: Lessons learned. *The journal of the learning sciences*, 4(2):167–207, 1995.

[3] S. Chaiklin. 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] B. Clement, P.-Y. Oudeyer, and M. Lopes. A comparison of automatic teaching strategies for heterogeneous student populations. In *EDM 16-9th International Conference on Educational Data Mining*, 2016.

[5] B. Clement, D. Roy, P.-Y. Oudeyer, and M. Lopes. Multi-armed bandits for intelligent tutoring systems. *JEDM-Journal of Educational Data Mining*, 7(n), 2015.

[6] J.-C. Falmagne, E. Cosyn, J.-P. Doignon, and N. Thiéry. The assessment of knowledge, in theory and in practice. In *Formal concept analysis*, pages 61–79. Springer, 2006.

[7] J. I. Lee and E. Brunskill. The impact on individualizing student models on necessary practice opportunities. *International Educational Data Mining Society*, 2012.

[8] A. N. Rafferty, E. Brunskill, T. L. Griffiths, and P. Shafto. Faster teaching by pomdp planning. In *AIED*, pages 280–287. Springer, 2011.

[9] S. Wang, F. He, and E. Andersen. A unified framework for knowledge assessment and progression analysis and design. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, pages 937–948. ACM, 2017.