Task and Timing: Separating Procedural and Tactical Knowledge in Student Models

Joshua Cook* , Collin F. Lynch, Andrew G. Hicks, & Behrooz Mostafavi
Department of Computer Science, North Carolina State University, Raleigh, NC, U.S.A.
jacook7, cflynch, aghicks3, & bzmostaf@ncsu.edu

ABSTRACT

BKT and other classical student models are designed for binary environments where actions are either correct or incorrect. These models face limitations in open-ended and data-driven environments where actions may be correct but non-ideal or where there may even be degrees of error. In this paper we present BKT-SR and RKT-SR: extensions of the existing BKT model that distinguish knowing *how* to apply a skill from knowing *when*. We compare their relative performance to that of classical BKT and PFA on data collected from Deep Thought, an open-ended propositional logic tutor. We develop basic performance curves for student outcomes to help us visually compare models predictions to data. We also introduce a new approach for finding a probability distribution of actions in ranked, multiple option environments with RKT and RKT-SR. Our results show that knowing when to use skills is more important than how in these open-ended contexts. In fact, including the *how* components may hurt performance if implemented naively. Furthermore we show that ranked models outperform binary-based models even under restrictive assumptions.

Keywords

Student-Modeling, Data-Driven Tutoring, Open-Ended Tutors, BKT, PFA, Interaction Networks, RKT, RKTSR, BKTSR

1. INTRODUCTION

Bayesian Knowledge Tracing (BKT) and other existing learner models, such as Performance Factors Analysis (PFA), are about *right* and *wrong* but for many realistic problem-solving situations students are not choosing just correct or incorrect actions. They are choosing from among a range of potential actions some of which may be optimal or substantively better than others. Thus the classical models are out of sync with the performance criteria by which the students are being judged. It also means that the models, by design, conflate two distinct skills: knowing *how* to apply a skill (procedural knowledge), and knowing *when* to apply a skill (tactical knowledge). In classical BKT we base performance on the validity of an individual action not on its optimality. Thus students receive points for correctly applying sub-optimal skills.

In this paper we present an extension to BKT, BKT-SR, which separates tactical knowledge (recognition of optimal skills) from procedural knowledge (correct skill application). This model is designed for use in open-ended and data-driven tutorial domains where students are expected to learn not just how to apply individual skills but how to recognize the sequence of skill applications that make up an optimal solution. We also present a refinement of the existing probability calculations for ranked options, and apply these in two new models: RKT and RKT-SR. This refinement leads to an improvement in the accuracy of the models over existing methods.

Additionally, in order to investigate which component of BKT-SR is most important, we tested the individual components (how, when, and some slight variations) on student data. Our data is drawn from an open ended propositional logic tutor called Deep Thought that is designed for use in discrete mathematics and philosophy. We compare the differing models on our data set to demonstrate that knowing when to apply a skill is separable from knowing how.

2. EXISTING MODELS

BKT and PFA are two of the most successful student modeling approaches. Both are binary action models that predict whether a student will take actions that are correct or incorrect at any given time given their level of understanding and other parameters. In prior head-to-head comparisons the two have performed similarly [5].

BKT is a simple two state Hidden-Markov Model (HMM) [3]. It is based upon five assumptions. Each skill is independent and has two states: learned, L, and not learned. Each problem depends on exactly one skill, and answers are either correct or incorrect. Students can learn, but cannot forget. After an opportunity to apply a skill, there is a constant probability to transition, T, from unlearned to learned. Students who know a skill will answer a problem correctly unless they slip, S, and students who don't know a skill answer incorrectly, unless they guess, G.

The parameters of BKT are: L0, the initial probability of knowing a skill. T, the probability of transitioning from unlearned to learned. G, the probability of answering a question correctly when a skill is not learned. S, the probability of answering a question incorrectly when a skill is learned.

*Corresponding author

Let L_i be the probability of knowing a skill at step i. Then the probability of answering a problem correctly is calculated as:

$$P(\text{Correct}) = L_i \cdot (1-S) + (1-L_i) \cdot G$$

To update L, we first apply Bayes theorem, then apply the transition probability. The reinforcement process has two steps:

$$B_i(\text{Answer}) = \begin{cases} \frac{L_i \cdot (1-S)}{L_i \cdot (1-S) + (1-L_i) \cdot G} & \text{Answer is correct} \\ \frac{L_i \cdot S}{L_i \cdot S + (1-L_i) \cdot (1-G)} & \text{Answer is incorrect} \end{cases}$$

$$L_{i+1} = B_i(\text{Answer}) + T \cdot (1 - B_i(\text{Answer}))$$

BKT is time tested, easily interpreted and implemented, but fitting BKT parameters is difficult. One difficulty lies in avoiding degenerate parameters: parameters that cause BKT to behave counter to its' physical interpretation. We avoid degenerate models using brute force grid search [5].

PFA, by contrast, is a logistic regression model based upon the skill difficulty(β), number of successes (γ), and number of failures(ρ) [11]. PFA has many upsides, not the least of which is that it can be fit efficiently with general regression calculations.

3. INTERACTION NETWORKS

The above models were designed for classical binary problems. Most realistic problems however are more open-ended. Problems are defined by a goal state and a set of given information that problem solvers may apply a range of rules to achieve their goal. Rather than each action being correct or incorrect some actions are correct in a given solution context and there many be many ways to solve a problem or many actions to take at a given time with some being more efficient than others. The structure of these open-ended solutions can be efficiently represented in a data structure called an interaction network. Interaction Networks are directed graphs representing a solution space where each node is a partial solution state and each edge is a rule application [4]. Individual solutions are represented as paths in the network from the start state to a goal state. An Interaction Network is the aggregation of all the student solutions for a problem where each edge is weighted by the number of students who followed it.

3.1 Value Iteration

Value iteration is an algorithm for identifying the optimal policy (π) for use in a Markov Decision Process (MDP) [1]. The core of the algorithm depends upon an update function that estimates the current value of a state ($V_{i+1}(s)$) based upon a set reward (R), the current values of the neighboring states ($V_i(u_e)$), a discount factor or cost for taking each action (γ), and the probability of taking an action ($P(e)$). In these experiments we use a constant reward function and a discount factor. Goal states were assigned a constant value, and the probability of a given action ($P(e)$) transitioning from state s to s' was estimated based upon the number of times that it was taken relative to the total number of steps out of s.

For the purposes of our study we defined two forms of the value function. The *optimistic* function assumes that students will take the best possible action in a given state and thus the best possible route to a goal. The *conservative* function, by contrast, assumes that they will follow the general probability distribution of the dataset. Thus:

Conservative: $V_{i+1}(s) = R + \gamma \cdot \sum_{e \in E_s} P(e) \cdot V_i(u_e)$

Optimistic: $V_{i+1}(s) = \max_{e \in E_s} R + \gamma \cdot P(e) \cdot V_i(u_e)$

The former approach was used in the Hint Factory system which uses interaction networks to generate data-driven hints [15], while the latter is equivalent to a single option MDP [16]. Any iteration that maximizes over contracting functions like these is, by definition, a contraction mapping [7]. Thus both forms will converge over time to a stable value.

4. OUR EXTENSIONS

We built several different extensions to the existing BKT model that are designed to take advantage of extra information in the interaction network to separate *tactical knowledge* (when to apply a skill) from *procedural knowledge* (how to apply a skill).

4.1 BKT-SR (BKT Skill Recognition)

BKT Skill Recognition (BKT-SR) is a semi-binary model that predicts students' behavior on a binary basis but reinforces on a more complex paired. In it we maintain two separate BKT models for each skill, one tracks procedural knowledge BKT_{How}, and the the other tracks tactical knowledge BKT_{When}. BKT-SR assumes that the ideal skill will be used only if the student correctly recognizes how to apply it, *and* knows that it is ideal.

The probability of answering a question correctly is the probability given by BKT_{How} multiplied by that given by BKT_{When}. The difference between the two models lies in their reinforcement. BKT_{How} reinforces the skill component of the action used, positively if it was used correctly. BKT_{When} reinforces skill component of both the action used AND the ideal action, positively if they are the same, negatively otherwise.

4.2 RKT (Ranked Knowledge Tracing)

Our environment is not binary, there are almost always several 'correct' options of ranked quality for each state. We therefore introduce the ranked models, RKT and RKT-SR. These models introduce a technique to give a probability distribution over a set of ranked options from simpler single skill models. The underlying model and reinforcement technique of RKT and RKT-SR is similar to BKT however it can be replaced by other comparable models so long as the reinforcement process is modified appropriately. This approach gives us a rigorous way to aggregate simple learner model predictions into a valid probability distribution over all actions. Conceptually, RKT tries the best option, if that fails it tries the second best, if that fails it tries the third and so on, wrapping back to the first.

Let x be our current model state and let $\alpha_i(x)$ be the probability that a student can use the skill required for option i given state x. Assuming the that the n skill options for a problem are given in order, the probability of using the i^{th} action is

$$p_i(x) = \frac{\alpha_i(x) \prod_{j=1}^{i-1} (1 - \alpha_j(x))}{1 - \prod_{j=1}^{n} (1 - \alpha_j(x))}$$

RKT's underlying model uses a simple two state Hidden-Markov Model (HMM) for each skill. State x is a vector of knowledge confidence. While $\alpha_i(x)$ is defined by taking the i^{th} component as L, and then calculating the probability as in standard BKT. RKT's update function is inspired by Bayes' theorem but differs slightly as our probability function is not linear. An exact, naïve imple-

mentation of an HMM would require that we sum over every combination of skill knowledge, which is prohibitively expensive.

To illustrate the update algorithm, suppose skill k is applied in state x, and that x_j is the probability of knowing skill j, and u_j is x with the j^{th} skill set to 1. We then calculate the new value for skill j, y_j, as:

$$y_j = \frac{p_k(u_j) \cdot x_j}{p_k(x)}$$

After each update we apply our transition function only to the ideal skill model. This function is applied in the same way as in BKT. Here p_i is convex in each argument, so our update will keep L between 0 and 1. Further, it will increase L iff knowing L will increase the chance of the given action. Thus the update is consistent and in the appropriate direction.

4.3 RKT-SR (RKT Skill Recognition)

Like BKT-SR, RKT-SR tries to separate procedural and tactical knowledge using two parallel RKTs, one for *how* and one for *when*. Like RKT, for state x, let $\alpha_i(x)$ denote confidence of being able to apply the skill used in option i, and $\beta_i(x)$ denote confidence of being able to identify when to use skill of option i.

In the RKT-SR approach we model the student's process as first noticing a set of options (how skill). Then, of the noticed options, they rank them (when skill). And finally they select the highest rank action to the best of their ability. Thus the probability of doing action i is:

$$p_i(x) = \sum_{\{i\} \in S \subseteq [n]} \frac{1}{1 - \prod_{j \in [n]} (1 - \alpha_j(x))} \prod_{j \in S} \alpha_j(x) \prod_{j \in [n] \setminus S} (1 - \alpha_j(x)) \cdot \frac{\beta_i(x) \prod_{j < i, j \in S} (1 - \beta_j(x))}{1 - \prod_{j \in S} (1 - \beta_j(x))}$$

This simplifies to:

$$p_i(x) = \frac{\alpha_i(x)\beta_i(x)}{1 - \prod_{k \in [n]} (1 - \alpha_k(x))} \sum_{j=0}^{\infty} (1 - \beta_i(x))^j \cdot \prod_{k=1}^{i-1} (\alpha_k(x)(1 - \beta_k(x))^{j+1} + 1 - \alpha_k(x)) \cdot \prod_{k=i+1}^{n} (\alpha_k(x)(1 - \beta_k(x))^j + 1 - \alpha_k(x))$$

Assuming that each β is bounded away from 1 and 0, we can approximate the infinite sum by taking a fixed number of terms, then normalizing it. For the sake of efficiency, we limit the number of terms to 3. We believe that RKT-SR has a convex probability function like RKT. Thus we update it similarly, with how and when updated independently.

Note that setting all $\alpha_i = 1$ in this model yields RKT, as does setting all $\beta_i = 1$. Thus RKT does not necessarily claim that either tactical or procedural knowledge is more important, since modelling either one with the assumption that the other is trivial yields the same model.

5. DATA SET

For this analysis we collected data from two semesters of an undergraduate Discrete Mathematics course at NCSU where Deep Thought is used. This dataset includes 4 class sections, 205 students, 2322 problem attempts, and 28640 individual steps. Unfortunately the data includes several cases where individual events were not logged such as the student entering or exiting the program, and cases where events were logged out of order due to network issues. While we cleaned these up as much as possible, we still include 913 errors in our data that we could detect but could not fix. While this missing data may contain important information, the average student only had a few such errors, even though 148 of the students had some kind of error in their logs.

In open-ended tutors like Deep Thought, problem-solving errors (i.e. incorrect applications) are often treated in one of two ways. Either the system records the mistake but leaves it onscreen and does not permit it to hinder forward progress. Or the system forces the student to fix or retract it immediately. In effect this forces the user to always step back to their prior state before moving on. Deep Thought adopts this latter approach. Consequently it is possible to ignore user mistakes in our dataset or to recognize them explicitly. With that in mind we tested our models with two different interaction networks. One network ignored self-loops, thus ignoring mistakes, and the other included them.

For each state, we ranked the set of derived statements to obtain a canonical order. Thus the states are dependant only on what was derived, not how or when it was derived.

5.1 Deep Thought

Deep thought is an intelligent tutoring system for propositional logic. Deep thought has been continually improved with hints [15], worked examples [10], and proficiency profiling [9]. The system's assessments have been verified against student test scores [8]. Deep Thought uses a GUI to guide students through 6 problem levels with increasing difficulty. Problems in Deep Thought are presented as a set of logical assumptions, and a statement which the student must to derive from them by applying axioms of propositional logic.

6. METHODOLOGY

We first generated the networks using all of the student data. This ensured that all actions taken by the students were included in the graph thus simplifying our analysis. This was not expected to bias in favor of any model. For the modeling step we only calculated the error and reinforced the models based upon steps with multiple correct options.

We used InVis to produce the graphs and perform the value iteration [12]. We fixed the value of our goal states at 100, used a negative immediate reward for each action of -1, and a discount factor of 0.9. Every other state started with a value of 0.

When measuring error, we focus on the cases where the system predicts that that a student will take the ideal action. We use a running average as our baseline. For the present we are more interested in the relative performance of our models than their absolute performance against chance.

In many states there are two distinct ideal actions that lead to different states with the same value. In this case, we want to know if a student completed either one. To get the appropriate probability of an ideal action we calculate the individual probabilities of the two ideal actions and, assuming they are independent, we then return the probability that either one is

performed. This approach works for simpler models like BKT and PFA which return per-action probabilities. However it may be unfairly penalizing RKT and RKT-SR, who return a complete probability distribution.

We tested our models using 10 fold cross validation. Each model was fit using an exhaustive grid search minimizing RMSE. Final metrics were found by calculating the RMSE and AUC for each fold, and then averaging them.

6.1 Applying Binary Models

BKT and PFA are not designed to handle non-ideal solutions, thus their models do not tell us how to reinforce them in this case. For each skill, we can reinforce the underlying knowledge component of the skill positively (*reward*), or negatively (*punishment*). Thus each model is seen as a black box, where we "select" skills to reinforce, and reward or punish it appropriately. In this context we can reinforce the sklls that the student actually performed as well as the ideal skills, which they may not. Here we tested four different versions of BKT which differ in what skills are selected for punishment and which are selected for reward.

Stock-BKT: This focuses solely on the students' demonstrated skills, ignoring idealness. It selects the skill used and rewards it if the action is correct. **ActualSkill-BKT:** This focuses on the students demonstrated skills, but with only the best possible action considered correct. It selects the skill used and rewards it if it is ideal. **IdealApp-BKT:** Focuses on whether or not the student knows which action is ideal and penalizes them for anything else. Selects the ideal skill and rewards if it was used ideally. The model makes no change if they performed a correct, but non-ideal use of the skill, and it punishes otherwise. **IdealActual-BKT:** Attempts to model both using a joint probability distribution. Thus it explicitly conflates knowing when to do something and knowing how and then sets a standard of correctness consistent with that. Selects both the ideal and the applied skills. If the ideal skill is used it is rewarded, otherwise both are punished.

We chose to reinforce PFA and the running average using the same selection model as in ActualSkill-BKT. For reference, BKT-SR is equivalent to IdealActual-BKT times Stock-BKT, reinforced independently.

6.2 Plotting Performance

In order to quickly check for skill acquisition, we developed a visualization technique. For each student, we look at the opportunities that they had to apply a skill ideally, and whether they actually used it. We then plotted these frequencies for all students on a single scatter plot.

Specifically, for each student x, and for each skill k, we make vector k^x, where the length of k^x is the number of times when skill k was ideal, with k_i^x 1 if the student used the ideal option the ith time k was ideal, 0 otherwise. Let $n_x(i)$ be the set of skills that were ideal at least i times. Define v^x as

$$v_i^x = \frac{\sum_{k \in n_x(i)} k_i^x}{|n_x(i)|}$$

Then we just plot each v^x together on a scatter graph. For comparison purposes we simulated data using BKT and plotted it using this technique. In it, you can see a clear trend. This trend is not clearly visible in our real data set. While some tweaking of the parameters in the simulated data show slower learning, they still show learning. Even graphs with errors look almost identical to the ones shown irrespective of value iteration algorithm. Thus this technique, while interesting, is ill-suited to detect learning in this domain.

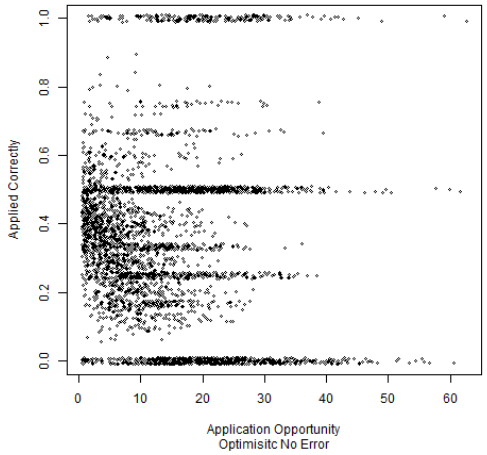

Figure 1: Real Data Performance

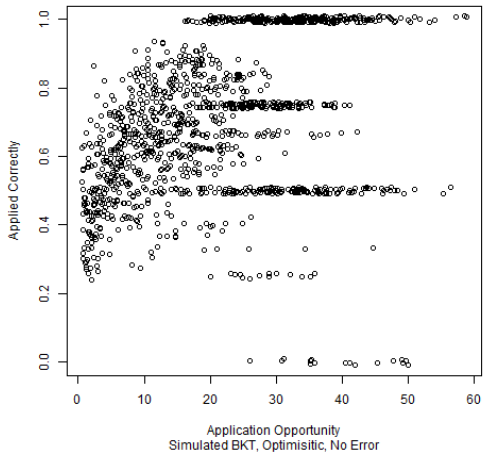

Figure 2: Simulated Performance

6.3 Model Fitting

We fit our parameters using exhaustive grid search. Grid search often performs favorably with other fitting methods like EM [14]. We define our grid by specifying the upper bound, the lower bound, and the number of equal length steps between them for each parameter. We chose the parameter bounds so that no fit would be degenerate [17]. BKT-SR used the same parameters to fit both the when and how subskills, but fits them independently to save time. Similarly for RKT-SR.

We chose the resolution for our grid search model in these cases to guarantee a similar amount of time per search, around 2

Table 1: Model Fitting Results

Model	Optimistic				Conservative			
	No Err		Err		No Err		Err	
	RMSE	AUC	RMSE	AUC	RMSE	AUC	RMSE	AUC
Average	0.451457	0.696120	0.438547	0.690875	0.465104	0.674632	0.446898	0.667558
PFA	0.454968	0.690093	0.442861	0.681035	0.469697	0.661166	0.451412	0.660922
Stock-BKT	0.493906	0.664096	0.489387	0.647382	0.492487	0.663387	0.495561	0.633865
ActualSkill-BKT	0.458204	0.676102	0.446208	0.671619	0.471135	0.656281	0.454614	0.646841
IdealApp-BKT	0.452686	0.699546	0.438583	0.709654	0.465627	0.681597	0.448043	0.686899
IdealActual-BKT	*0.449347*	0.697695	*0.436518*	0.704180	*0.462758*	0.682124	*0.444161*	0.684025
BKT-SR	0.452071	0.691284	0.469820	0.650012	0.465264	0.671495	0.479585	0.628389
RKT	0.450763	*0.737032*	0.437331	*0.724183*	0.464668	*0.709409*	0.447027	*0.704591*
RKT-SR	**0.440841**	**0.739516**	**0.432296**	**0.729586**	**0.455561**	**0.715869**	**0.438965**	**0.713305**

Table 2: KT Fitting Parameters

	BKT				BKTSR				RKT				RKTSR			
	L0	T	G	S	L0	T	G	S	L0	T	G	S	L0	T	G	S
Min	0.1	0.02	0.04	0.02	0.2	0.03	0.04	0.03	0.2	0.06	0.07	0.06	0.3	0.06	0.08	0.1
Steps	5	5	5	5	4	4	4	4	3	4	4	3	2	3	3	2
Max	0.9	0.30	0.40	0.30	0.8	0.30	0.40	0.30	0.8	0.30	0.40	0.30	0.7	0.30	0.40	0.25

Table 3: Baseline Fitting Parameters

	Running Avg		PFA		
	Prior Avg	Start	β	γ	ρ
Min	0.00	1	-2.4	0.05	-1.25
Steps	21	21	9	9	9
Max	1.00	101	2.4	1.25	-0.05

hours, save for RKT-SR, which takes about 5 times as long as RKT to run, and takes 10 times as long to fit using our grid search. We determined that lowering the resolution any more would make fitting ineffective. We expect that the real running time could be greatly improved through code tweaks and by using a more efficient implementation language.

7. RESULTS

The results of the optimistic and conservative value iteration are largely equivalent, with every model predicting a little better on the optimistic value iteration, including the running average. This is likely because the optimistic value iteration favors the most frequently used options more than conservative value iteration.

Stock-BKT, the standard *how* BKT, performed worse then any other model across the board. This implies that tactical knowledge is more important then procedural knowledge in this domain. Surprisingly, removing all error observations does not change the performance of Stock-BKT relative to the other models.

ActualSkill-BKT does slightly worse then a running average, as does PFA, but IdealApp-BKT, which reinforces the ideal skill alone, performs better, trading blows with the running average. This suggests that using the wrong skill is more an indication that the right skill is not known, rather than that the used skill is unknown. Ultimately it appears that they are more important together, this is supported by the fact that IdealActual-BKT outperforms both the other models and the running average.

BKT-SR does not perform as well as its *when* sub-component, IdealActual-BKT. In fact, when we include errors in our data set, BKT-SR does significantly worse. The fact that including errors did not help Stock-BKT or BKT-SR was a surprise. This seems to suggest that failing to use a skill correctly does not always stem from not knowing that skill. We suggest that this is actually just noise from random guesses. When looking at individual records, we find that this is consistent with what we have seen in the logs. There we find long stretches where students solve problems in order followed by bursts of failed skill applications. Thus the extra noise in the *how* component of BKT-SR hurts the model.

But, if we compare the more informed models, RKT and RKT-SR, we get a better picture. RKT-SR is the best performing model across the board with RKT second in terms of AUC, and IdealActual-BKT second in RMSE. RKT and RKT-SR incorporate more then just the ideal option, their predictions incorporate all of the other skills into the probabilities. Thus in BKT terms, the guess and slip are not constant, and they depend upon the other options and upon how good the student is with them. In line with this, RKT and RKT-SR reinforces every applicable skill, not just a few.

Both RKT and RKT-SR assume that the options are ordered, the conceptual difference is that RKT does not distinguish between procedural and tactical knowledge. That is enough to outdo all our other models (except RKT-SR) in terms of AUC. Unlike our simpler models, incorporating both how and when information further improves performance, as RKT-SR outperforms RKT. So *when* and *how* are both different and useful concepts, but separating them takes a little more effort then BKT-SR.

8. CONCLUSIONS & FUTURE WORK

Open-ended tutoring systems are designed to teach students not only how to apply a skill but when to do so. Classical student modeling approaches, however, have focused entirely on procedural knowledge and generally ignore tactical information. In practice it is often difficult to assess whether or not students

are gaining this tactical knowledge and prior studies have either assumed it or have been content to conflate the two.

In this paper we address this lack of information in two ways. First we sought to visually inspect improvements in tactical knowledge. We found that for real student data there is no clear or statistically significant indication of improvement. We therefore opted to develop novel student models that incorporate this information and then to assess their performance on real student data.

In future work we plan to enhance the structure of both our experimental and baseline models. Since this project started, there have been a number of interesting extensions to BKT, such as adding forgetting, and latent student abilities [6]. We did not implement these extensions, but they should be directly applicable to this context, as well as to RKT and RKT-SR.

Additionally, Deep thought originally implemented interaction networks for the purposes of hint generation [15]. Later improvements saw worked examples incorporated into it [10]. This significantly effected student behaviour. Since none of our models integrate contextual data, we restricted our data to the students that saw no worked examples. In future, we may modify the update for the model to incorporate the worked examples. This integration of contextual information has been done before [13], but in this case it is probably more accurate to apply a transition probability.

Many interactive tutors have solutions that can be expressed as an interaction network and thus can be used with these methods. These include Andes [18], and Pyrenees [2]. We will seek to generalize these results by testing them on datasets collected from these tools.

RKT and RKT-SR are new models which make strong assumptions. In future work we will reevaluate the behavior of these models and the underlying assumptions that they make. RKT, for example, assumes that quality is ranked, but removing that assumption could change the model significantly.

RKT gives a valid probability distribution over all options, but we have only tested its accuracy in predicting whether the ideal action is used. We did not test whether or not it was accurate at predicting which of the other actions would be used. This is believed to be an advantage of RKT, but we have not verified that.

9. ACKNOWLEDGMENTS

This research was supported in part by the Provost's Professional Experience Program (PEP) at North Carolina State University.

10. REFERENCES

[1] R. Bellman. A Markovian Decision Process. *Journal of Mathematics and Mechanics*, 6, 1957.

[2] M. Chi and K. VanLehn. Meta-cognitive strategy instruction in intelligent tutoring systems: How, when, and why. *Educational Technology & Society*, pages 25–147, 2010.

[3] A. T. Corbett and J. R. Anderson. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User modeling and user-adapted interaction*, 4:253–278, 1995.

[4] M. Eagle, M. Johnson, and T. Barnes. Interaction networks: Generating high level hints based on network community clustering. In *Proceedings of the fifth international conference on educational data mining*, pages 164–167, 2012.

[5] Y. Gong, J. E. Beck, and N. T. Heffernan. Comparing knowledge tracing and performance factor analysis by using multiple model fitting. In *Intelligent Tutoring Systems: 10th International Conference*, pages 35–44, 2010.

[6] M. M. Khajah, R. V. Lindsey, and M. C. Mozer. How deep is knowledge tracing. In *Proceedings of the 9th International Conference on Educational Data Mining*, pages 94–101, 2016.

[7] J. E. Marsden and M. J. Hoffman. *Elementary Classical Analysis*. W.H. Freeman and Company, 1993.

[8] B. Mostafavi and T. Barnes. Exploring the impact of data-driven tutoring methods on students' demonstrative knowledge in logic problem solving. In *Proceedings of the 9th International Conference on Educational Data Mining*, pages 460–465, 2016.

[9] B. Mostafavi, Z. Liu, and T. Barnes. Data-driven proficiency profiling. In *Proceedings of the 8th International Conference on Educational Data Mining*, pages 335–341, 2015.

[10] B. Mostafavi, G. Zhou, C. Lynch, M. Chi, and T. Barnes. Data-driven worked examples improve retention and completion in a logic tutor. In *17th International Conference on Artificial Intelligence in Education*, pages 726–729, 2015.

[11] Philip I. Pavlik Jr., H. Cen, and K. R. Koedinger. Performance factors analysis âĂŞ a new alternative to knowledge tracing. In *Proceedings of the 2009 conference on Artificial Intelligence in Education*, pages 531–538, 2009.

[12] V. Sheshadri, C. Lynch, and T. Barnes. Invis: An edm tool for graphical rendering and analysis of student interaction data. In *EDM 2014 (G-EDM 2014: Workshop on Graph-based Educational Data Mining)*, pages 65–69, 2014.

[13] R. S.J.d. Baker, A. T. Corbett, and V. Aleven. More accurate student modeling through contextual estimation of slip and guess probabilities in bayesian knowledge tracing. In *Proceedings of the 9th International Conference on Intelligent Tutoring Systems*, pages 406–415, 2008.

[14] R. S.J.d. Baker, Z. A. Pardos, S. M. Gowda, B. B. Nooraei, and N. T. Heffernan. Ensembling predictions of student knowledge within intelligent tutoring systems. In *Proceedings of the 19th international conference on User modeling*, pages 13–24, 2011.

[15] J. C. Stamper, M. Eagle, T. Barnes, and M. Croy. Experimental evaluation of automatic hint generation for a logic tutor. In *Artificial Intelligence in Education*, pages 345–352, 2011.

[16] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press Cambridge, 1998.

[17] B. Van De Sande. Properties of the bayesian knowledge tracing model. *Journal of Educational Data Mining*, 5(2):253–278, 2013.

[18] K. Vanlehn, C. Lynch, K. Schulze, J. A. Shapiro, R. Shelby, L. Taylor, D. Treacy, A. Weinstein, and M. Wintersgill. The andes physics tutoring system: Lessons learned. *International Journal of Artificial Intelligence in Education*, pages 147–204, 2005.