Improving Student Modeling Through Partial Credit and Problem Difficulty

Korinn Ostrow, Christopher Donnelly, Seth Adjei, Neil Heffernan
Worcester Polytechnic Institute
100 Institute Road
Worcester, MA 01609
{ksostrow, cdonnelly, saadjei, nth}@wpi.edu

ABSTRACT
Student modeling within intelligent tutoring systems is a task largely driven by binary models that predict student knowledge or next problem correctness (i.e., Knowledge Tracing (KT)). However, using a binary construct for student assessment often causes researchers to overlook the feedback innate to these platforms. The present study considers a novel method of tabling an algorithmically determined partial credit score and problem difficulty bin for each student's current problem to predict both binary and partial next problem correctness. This study was conducted using log files from ASSISTments, an adaptive mathematics tutor, from the 2012-2013 school year. The dataset consisted of 338,297 problem logs linked to 15,253 unique student identification numbers. Findings suggest that an efficiently tabled model considering partial credit and problem difficulty performs about as well as KT on binary predictions of next problem correctness. This method provides the groundwork for modifying KT in an attempt to optimize student modeling.

Author Keywords
Student modeling; tabling method; partial credit; problem difficulty; next problem correctness; knowledge tracing

ACM Classification Keywords
K.3.1 Computer Uses in Education, I.2.6 Learning, I.6.4. Model Validation and Analysis

INTRODUCTION

Modeling student learning within an intelligent tutoring system can be a daunting task. In order to make predictions about a student's knowledge or their next problem correctness, models must decipher noisy input and isolate only those features that define the probability of knowledge or learning. As such, designers of intelligent tutoring systems have largely relied on Knowledge Tracing (KT), as presented by Corbett & Anderson [2], to model the probability of student learning at real time within popular systems such as Cognitive Tutor [5]. Other methods, such Performance Factors Analysis, seek to model learning when considering overlapping knowledge components (i.e., skills) and individualized student metrics [11], offering an alternative to KT in certain circumstances.

Despite the popularity of KT and PFA, the standard models rely on binary input to establish predictions of students' knowledge state or performance, failing to consider continuous metrics that would better individualize the model across students or skills. Expansion in the field educational data mining has since lead to a number of alternative or supplementary learning models. For instance, researchers have attempted to impart individualized prior knowledge nodes for each student [8], to supplement KT with a flexible metric for item difficulty [9], to ensemble various methods of binning student performance (i.e., partial credit) with standard KT models [12], and to consider the sequence of a student's actions within the tutor to help predict next problem correctness [3].

Without modifying KT or PFA directly, adding parameters to student learning models can be a limited approach. Tabling methods that quickly establish maximum likelihood probabilities have previously been used by Wang & Heffernan [12, 13] to test and optimize various potential adaptations to KT. Following in this process, the present study uses a tabling method to lay the groundwork for future modifications to KT that will allow for predictions of next problem correctness using the partial credit score and difficulty estimate of the current item. While previous work has shown the benefit of ensembling tabling methods with KT, we hope to use the findings presented herein to modify KT directly, as it has previously been suggested that ensembling can be a rather sensitive approach [4].

Perhaps standard learning models rely on binary correctness as measured by a student's first response at each skill opportunity (i.e., a sequence of correct and incorrect responses based on a student's first action within each problem) due to the complexity of accurately and universally defining an algorithm that validates partial credit scores within intelligent tutoring systems. Within the

L@S 2015, March 14–18, 2015, Vancouver, BC, Canada.

.http://dx.doi.org/10.1145/2724660.2724667

majority of current learning models, a student would be penalized with a score of zero for taking advantage of the tutoring that plays an integral role in these platforms. Yet the primary goal of most intelligent tutoring systems is not solely to assess student knowledge, but to simultaneously promote student learning through adaptive feedback, making binary correctness a stale concept. Students often require multiple attempts to solve a problem or request system feedback for guidance, thus assigning value to the concept of partial credit. Attali and Powers [1] suggested the benefits of considering partial credit when predicting learning outcomes in adaptive environments, as evidenced by the modification of standardized tests to allow partial credit when predicting GRE scores.

Within ASSISTments, an adaptive mathematics tutor, a naïve model of partial credit scoring was previously established by Wang and Heffernan [12], termed the "Assistance" Model. This method calculated maximum likelihood probabilities for next problem correctness using a twelve-parameter table built from binning students' hint usage and attempt count. In this manner, the authors used system features to indirectly gauge a partial credit metric that would help predict binary performance.

The present study provides methodological evidence that student modeling can be enhanced through the use of algorithmically derived partial credit scores and a binned metric of problem difficulty. We first use tabling method (a probabilistic approach employing maximum likelihood estimations) that considers the partial credit score of the current problem to predict both binary and partial next problem correctness. We also establish a more complex prediction table that considers both partial credit and problem difficulty. Through this novel concept, we hope to show that students can ultimately gain knowledge from a problem even if they fail to earn full credit. Our findings argue for the design of a modified KT model that is sensitive to a continuous measure of partial credit rather than binary input, and that isolates a known level of problem difficulty for each question. We seek to answer the following research questions:

1. Does an algorithmically determined partial credit score outperform binary metrics when used to predict next problem correctness?
2. Does a binned metric of current problem difficulty (e.g., Low, Medium, or High difficulty) provide a valid prediction of next problem correctness?
3. Can current problem difficulty supplement partial credit score to outperform similar modeling techniques?

DATASET

The dataset used for this analysis was compiled from problem logs from the ASSISTments platform during the 2012-2013 school year. The original file included roughly 1.5 million rows of problem level data (i.e., each row detailed all logged actions for one problem for one student). For this study, we chose to analyze only the top ten most densely populated knowledge components. Attributes of these skills are further explained in Table 1. The dataset examined here has been made publicly available at [7].

In order to properly calculate partial credit, approximately 5,000 rows were removed due to a lack of logged end time, meaning that these problems had never been properly completed. Using the platform's current grading method, which is based on the students' first response, these logs carry binary correctness scores. However, as the problem was ultimately considered incomplete, partial credit could not be determined with certainty and the logs were therefore excluded from analysis. Further, the analysis presented herein reports only on main problems. Scaffolding problems, a feedback style within the ASSISTments platform typically used to break a problem down into steps or to provide worked examples, were excluded from the final dataset. The decision to work with main problems was based in part on the justification made by Pardos & Heffernan [9] when using a similar dataset from the ASSISTments platform. As scaffolding problems are guided, they offer a less accurate view of skill knowledge and skew performance data within an opportunity based analysis. An analysis of the remaining dataset revealed that only 0.3% of first actions were scaffold requests, further supporting the intuition that the removal of scaffolding data was appropriate.

Due to the time constraints involved in running multiple models with five fold cross-validation (explained further in the Compared Models and Model Testing and Training sections), we chose to restrict the dataset to a maximum of 15 opportunities per student per skill. This reduced the dataset by 46,680 rows, primarily removing students who were excessively struggling and those gaming the system; the majority of students were unaffected by this refinement.

The resulting dataset consisted of 338,297 problem logs representative of 15,253 unique student identification numbers. On average, each student identifier linked to approximately 3.3 skills. Further exploration of this dataset revealed that it was comprised of 7,363 unique problems. A total of 3,787 unique assignments were made by 417 teachers spanning 231 schools. The skill content ranged from grades 6-8 as shown in Table 1. The majority of logged problems (over 90%) were completed by students who 'mastered' or finished the full assignment from which the problem originated.

Three types of questions were represented in the dataset. The majority of problems logged, 84.3%, were 'mathematical expressions,' a problem type that accepts any answer that is mathematically equivalent to the correct answer (i.e., answers of 1/2 and 0.5 are both accurate). In contrast, 12.5% of problems logged were 'fill-in,' a problem type that requires the student to input an *exact* string matching the preset correct response (i.e., if 1/2 was the preset answer, 0.5 would be incorrect). The remaining 3.2% of problems logged in the dataset were 'multiple choice,' featuring two or more answers available for selection.

Skill ID	Definition	Grade Level	# Logs (Rows)	% Resulting Dataset
277	Addition and Subtraction of Integers	7	44,731	13.2
311	Equation Solving with Two or Fewer Steps	7	44,005	13.0
280	Addition and Subtraction of Fractions	6	42,550	12.6
276	Multiplication and Division of Positive Decimals	6	37,033	10.9
47	Conversion of Fractions, Decimals, and Percentages	6	32,741	9.7
67	Multiplication of Fractions	6	31,716	9.4
61	Division of Fractions	6	28,809	8.5
278	Addition and Subtraction of Positive Decimals	6	27,301	8.1
310	Order of Operations	8	25,132	7.4
79	Proportions	7	24,279	7.2

Table 1. Skill details and distribution in resulting dataset

Further assessment of students' responses provided insight into their first actions, attempt counts, and hint usage. For 95.5% of logged problems, the student's first action was to make an answer attempt. Using ASSISTments' current scoring scheme, these attempts would receive binary scores of either correct (1) or incorrect (0). Within this subgroup of logged problems, 24% of the problems were marked as incorrect while 76% were marked as correct. This suggests that a partial credit metric could provide benefit for approximately one quarter of attempted questions. Of the remaining logged problems, 4.2% represented first action hint requests, and 0.3% represented first action scaffolding requests.

Given that partial credit scores for the present study are algorithmically derived from an assessment of the student's attempt count and hint usage for each logged problem, these variables were examined thoroughly. Analysis of attempt counts across logged problems revealed a minimum of 0 and a maximum of 496, with a mean of 1.47 and a standard deviation of 2.23. For logs that were marked as incorrect based on first action, mean attempts rose to 2.70 with a standard deviation of 3.99. Within the full dataset, students made a total of 496,533 attempts.

Hint counts were also analyzed across logged problems and compared to the total number of hints available for each problem. Each problem had at least one hint, usually serving as the bottom out hint (i.e., it provided the answer). The average number of hints available per problem was 3.38, with a standard deviation of 0.88. The majority of problems had three hints (38.9%) or four hints (33.4%), with the maximum number of hints available in any problem topping off at seven. Across all logged problems, a total of 1,090,225 hints were available. Of the available hints, students only used a total of 167,371, or roughly 15.4%. The average number of hints used was 0.49 with a standard deviation of 1.20. For problem logs in which students answered incorrectly on their first attempt, 55.8% of available hints were utilized. Information particular to the bottom out hint showed that within problems initially answered incorrectly, only 14.5% of students proceeded to the bottom out hint. Thus, when struggling, the majority of students used the adaptive feedback inherent to the tutoring system in an appropriate manner. This provides further evidence for consideration of valid partial credit metrics.

Figure 1 provides a screenshot of a typical problem within the ASSISTments tutor. Specifically, this problem is a representation of the second most densely populated skill in the 2012-2013 ASSISTments log file: "Equation Solving with Two or Fewer Steps." This skill is exemplified, rather than highlighting the top skill, "Addition and Subtraction of Integers," as the problem provides a more robust example of the system's tutoring feedback. As shown in Figure 1, the student is presented with the equation and asked to solve for the missing variable. He or she can make an attempt to solve the problem, or may ask for the first of three hints. The hints increase in specificity, in an attempt to guide the student without providing excess assistance. The first hint shown in Figure 1 provides a worked example of a similar problem solving for the missing variable, x. If the student is unable to proceed using only the worked example, he or she can request the second and third hints as needed. The third hint in Figure 1 is the bottom out hint; it provides the correct answer ("-24") in an attempt to keep the student from getting stuck in the assignment, as it is not possible to skip problems and return at a later point as one can with traditional bookwork.

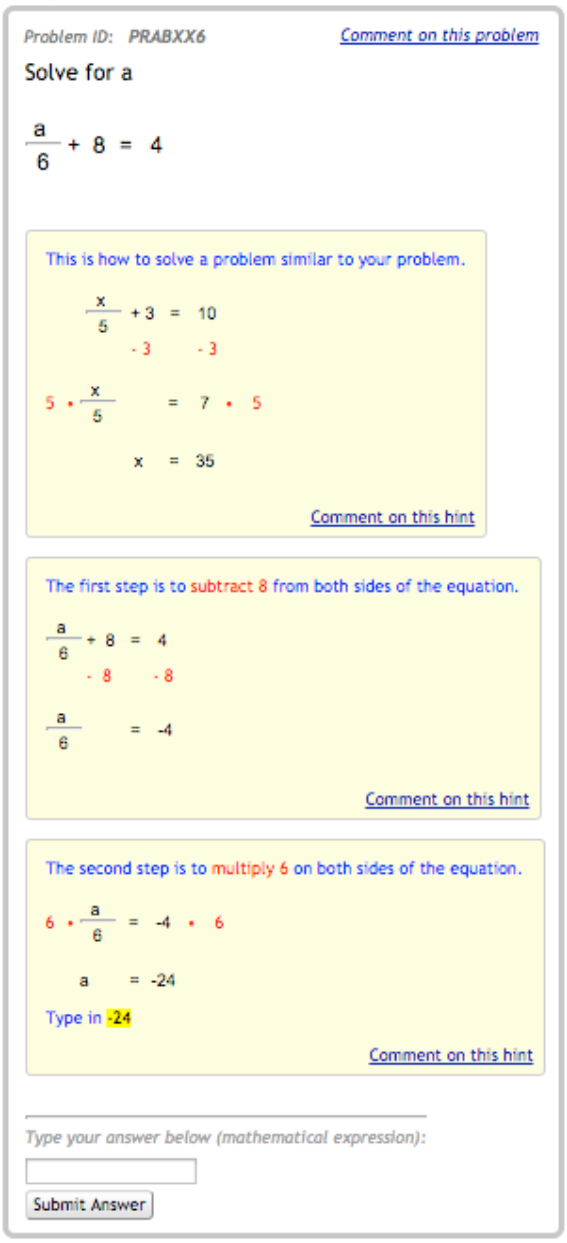

Figure 1. An example problem featuring three hints for the skill "Equation Solving with Two or Fewer Steps"

COMPARED MODELS

The following subsections explain the design and brief history (when appropriate) of the five models compared in the current study. All five models are primarily designed to predict binary next problem correctness. For permitting models, we present predictions of partial credit next problem correctness using continuous probabilities for additional consideration.

Partial Credit Predicting Next Problem Correctness

A naïve partial credit algorithm was derived by the ASSISTments design team in hopes of providing the system with partial credit scoring capabilities based on students' attempt count and feedback usage. Scores were determined subjectively based on teacher input and a conceptual understanding of how students typically behave within the tutoring platform. For this study, the algorithm was altered slightly to consider multiple problem types and to account for the students' first action. For instance, if a student asked for tutoring feedback without making an attempt to solve the problem, we felt that a larger penalty was merited.

The resulting algorithm used to define partial credit scores is depicted in Figure 2. Rather than establishing a deduction method on a per hint or per attempt basis as shown in previous work [12], the algorithm presented in Figure 2 places each logged problem into one of five partial credit bins (0, .03, 0.6, 0.7, 0.8, 1.0) by considering the logged data pertaining to first response type (attempt = 0, hint request = 1, scaffold request = 2), attempt count, and hint count.

For example, if a student makes only one attempt and is correct without requiring feedback, they earn full credit (a score of 1). This is similar to the notion of binary correctness on first response that ASSISTments currently employs. However, in the current method, all other first actions equate to an incorrect answer (i.e., requesting a hint or scaffold, or making a first attempt that is incorrect would both earn the student a score of 0).

As shown in Figure 2, after ruling out a 'correct' response, the partial credit algorithm considers whether the student requested a scaffold before even making an attempt. This behavior would suggest that either the student was not actually trying to answer the problem, or that he or she was struggling conceptually. Thus, rather than earning no credit, the student is only discounted to a score of 0.6.

```
IF type = algebra OR type = fill_in
   IF attempt = 1 AND correct = 1 AND hint_count = 0
      THEN 1
   ELSIF first_action = 2
      THEN .6
   ELSIF attempt < 3 AND hint_count = 0
      THEN .8
   ELSIF (attempt <= 3 AND hint=0)
      OR (hint_count = 1 AND bottom_hint != 1)
      THEN .7
   ELSIF (attempt < 5 AND bottom_hint != 1)
      OR (hint_count > 1 AND bottom_hint != 1)
      THEN .3
   ELSE 0

IF type = multiple_choice
   IF correct = 1
      THEN 1
   ELSE 0
```

Figure 2. Algorithm used to determine Partial Credit score based on first response, attempt count, and hint usage

Regardless of the student's first action, if he or she uses less than three attempts and does not request any hints, they earn slightly more, with a score of 0.8. The next bin is marked by students who have three or fewer attempts and have not used a hint, or those who have asked for only one hint and were not provided the answer (i.e., if a student's first action is to request a hint that is not the bottom out hint, they would fall into this bin). These students earn a score of 0.7. If the student can solve the problem within 5 attempts without seeing the bottom out hint, or if he or she uses multiple hints without ultimately reaching the bottom out hint, their partial credit score is 0.3. Finally, for students who use five or more attempts, or for those that see the answer, the problem is marked incorrect (a score of 0).

For multiple-choice questions the algorithm reverts to binary correctness because this type of problem does not usually provide feedback and guessing can be far more prevalent and consequential. Thus, if a student fails to get the correct answer on their first attempt, he or she receives a score of 0. This method was employed to keep the problem type from gaining an unfair advantage within the dataset. For instance, using the algorithm applied to other problem types, a student guessing through a multiple-choice problem with only four responses would still receive a score of 0.3.

The full algorithm was run across the dataset and partial credit scores were obtained for each logged problem. These partial credit scores were then used to define a pivot table to predict averages for both binary and partial next problem correctness, using maximum likelihood estimation. Results are presented in Table 2. For all parameter Tables, the number of logged problems falling into respective bins is depicted by sample size, *n*. The distribution of the data suggests that slight improvements could be made to the partial credit algorithm as few students fell into the 0.6 bin. Of all available 'next problem' data, only 14.7% of logs had partial credit values between 0 and 1. Thus, 85.3% of students would be insured by the platform's current method of binary correctness. This suggests that any significant finding among the models considered in the present study would be quite intriguing, as only a small portion of the sample is actually receiving the 'partial credit' treatment.

Partial Credit	n	Binary	Partial
0	45,735	0.5062	0.5634
0.3	6,471	0.5902	0.7438
0.6	940	0.3660	0.7948
0.7	12,077	0.6921	0.8396
0.8	22,797	0.7085	0.8668
1	200,287	0.8050	0.8785

Table 2. Parameters for predicting Binary and Partial Next Problem Correctness from current problem Partial Credit

It should be noted that a potential problem inherent to this tabling method (apparent in all tabled models in the present study) is the inability to predict correctness on a student's first opportunity within a skill, as there is no preceding problem data. This essentially causes the loss of 49,990 rows of data representing first problem predictions. Thus, sample sizes in Tables 2, 3, and 4 total 288,307 logs rather than 338,297.

Problem Difficulty Predicting Next Problem Correctness

A continuous metric of problem difficulty was calculated by retrieving data from all problems logged in the platform that were created before August 2012 (i.e., prior to the first timestamp in the modeling dataset). For each unique problem, all existing logs were averaged and a percentage of correct responses were determined. The resulting value offers an inverse metric of the problem's difficulty level. For instance, a problem on which students averaged 80% on all previous opportunities would not be considered very difficult. This metric was then binned into Low, Medium, and High difficulties by defining Medium difficulty as scores falling within +/- 0.5 standard deviations from the mean. Considering the inverse nature of the metric, High difficulty problems therefore had continuous values *below* this cut off, and Low difficulty problems had continuous values *above* this cutoff.

The bins for current problem difficulty were used in a maximum likelihood probability table to predict averages for both binary and partial scores for next problem correctness. Resulting parameters are presented in Table 3.

Difficulty	n	Binary	Partial
Low	91,712	0.7764	0.8465
Medium	107,901	0.7452	0.8297
High	88,694	0.6928	0.7895

Table 3. Parameters predicting Binary and Partial Next Problem Correctness from current problem Difficulty

Partial Credit and Problem Difficulty Predicting Next Problem Correctness

Based on the definitions of partial credit and problem difficulty defined in the singular models above, our goal was to create a novel model that used a tabling approach to consider partial credit together with problem difficulty to make predictions about next problem correctness. For each logged problem, partial credit score and problem difficulty were referenced to determine parameters for both binary and partial credit next problem correctness. Resulting probabilities are presented in Table 4.

		High			Medium			Low	
Partial Credit	**n**	**Binary**	**Partial**	**n**	**Binary**	**Partial**	**n**	**Binary**	**Partial**
0	8,357	0.5130	0.5621	16,307	0.5027	0.5622	21,071	0.5062	0.5650
0.3	1,107	0.6035	0.7401	2,332	0.6017	0.7548	3,032	0.5766	0.7367
0.6	29	0.5902	0.8508	236	0.3388	0.7897	675	0.3661	0.7943
0.7	2,829	0.6971	0.8288	4,888	0.6987	0.8463	4,360	0.6816	0.8391
0.8	5,094	0.7770	0.8753	8,342	0.7354	0.8712	9,361	0.6473	0.8581
1	74,296	0.8116	0.8787	75,796	0.8072	0.8841	50,195	0.7921	0.8697

Table 4. Parameters predicting Binary and Partial Next Problem Correctness from Partial Credit and Problem Difficulty

Knowledge Tracing

Knowledge Tracing (KT) is perhaps the most common method for modeling student performance. The standard KT model [2] has successfully proven itself as the basis for modeling student knowledge within intelligent tutoring systems [5] and thereby serves as a stable comparison for new work.

As shown in Figure 3, the standard model of KT is a Bayesian Network comprised of four learned parameters. Two parameters represent student knowledge (*prior knowledge* and *learn rate*) and two parameters represent student performance (*guess rate* and *slip rate*). The standard KT model is binary in that skills can only be in a 'learned' or 'unlearned' state, and questions can only be 'correct' or 'incorrect.' The model is updated with each skill opportunity based on the student's performance by using the following equation as defined by Corbett & Anderson [2]:

$$p(L_n) = p(L_{n-1}|evidence) + \left(1 - p(L_{n-1}|evidence)\right) * p(T)$$

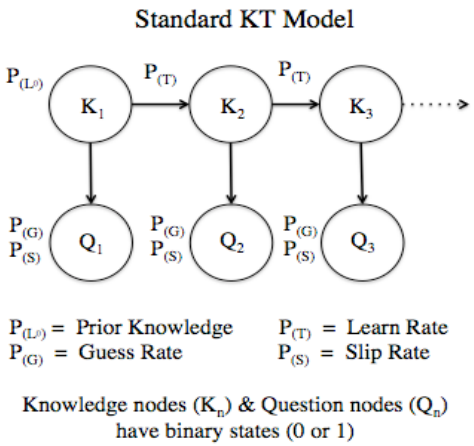

Figure 3. The standard Knowledge Tracing model with all learned parameters and nodes explained

Forgetting does not factor into the standard KT model when observing individual skills, as *guess* and *slip* parameters are thought to account for incorrect answers within the students' sequence of opportunities. For further information regarding the details of KT, refer to [2].

For this study, KT analysis was performed using the Bayes Net Toolbox (BNT), a popular open-source code for fitting directed graphical models within MATLAB [6].

Performance Factors Analysis

Performance Factors Analysis (PFA) was proposed as an alternative to KT by Pavlik, Cen, and Koedinger [11]. The method can model problems with multiple skills and has been shown to accurately model and select practice within adaptive systems. PFA was derived from Learning Factors Analysis (LFA), an approach that considers a parameter for student ability, a parameter for the skill's difficulty, and a learning rate for each skill. While PFA still considers skill difficulty, β, the model improves upon LFA by considering the frequency of both correct and incorrect answers in a student's response pattern, rather than simply assessing the frequency of skill practice. Thus, PFA predictions are updated with each skill opportunity based on a cumulative history of the student's successes (weighted by γ) and failures (weighted by ρ), as depicted in the following equation defined by Pavlik, Cen, and Koedinger [11]:

$$m(i, j \in KCs, s, f) = \sum_{j \in KCs} (\beta_j + \gamma_j s_{i,j} + \rho_j f_{i,j})$$

The log-likelihood (m) attained through this equation can then be passed through an exponential function to find the probability that the student will get the item correct. This model suggests that learning is defined by more than just skill practice, and that performance is strongly tied to skill acquisition.

For this study, PFA was performed using unpublished code within MATLAB. With properly formatted data, the analysis can also be performed using logistic regression in common statistical packages like IBM's SPSS.

MODEL TRAINING AND TESTING

Five-fold cross validation was used to train and test each model. In order to perform five-fold cross validation within our tabled models, the dataset was divided using a modulo operation on each student's unique identification number. Thus, for every student in the file, student id mod 5 was called, returning a remainder falling into bins from 0 to 4, thereby assigning students to folds. The distribution of the resulting folds was roughly equivalent, as shown in Table 5. With 15,253 unique student identification numbers in the dataset, the largest fold had 3,082 student ids and the smallest fold had 2,996 student ids, leaving a range of 86 and a standard deviation of 33.7.

Within each iteration of the cross-validation process, the model was trained on approximately 80% of the data and tested on the 20% that had been held out. Thus, when trained on folds 1, 2, 3, and 4 (80% of the data) the model would impart predictions on fold 0 (the held out 20%). In this manner, for each tabling method described in Section 3, table parameters were learned using four training folds and predictions were made on the held out fold. The process was repeated for all folds, thus resulting in five probability tables for each prediction type (i.e., five 'training' tables for partial credit predicting binary next problem correctness). Using an extensive formula in Microsoft Excel, the predicted averages were then applied back to each logged problem respective of test fold. For predictions of binary next problem correctness, rather than arbitrarily selecting a cutoff point for classifying binary correctness (e.g., simply using values greater than 0.5 to convey '1'), we instead subtracted the prediction directly from the actual binary result. Thus, when predicting next problem correctness using partial credit alone, if the next problem is actually correct using binary standard, the resulting residual is calculated as: 1.0000 - 0.7085 = 0.2915. In this manner, residuals were calculated for each log entry in each test fold that contained data for next problem correctness.

A similar method of five fold cross-validation was coded into the KT and PFA analyses within MATLAB. Without modification, KT and PFA are not intended to accurately predict partial credit next question correctness, and as such we have saved these analyses for future work.

Fold	Unique Students	# Logs	% Dataset
0	3074	67,715	20.0
1	3046	68,081	20.1
2	3082	68,739	20.3
3	3054	67,996	20.1
4	2996	65,766	19.4

Table 5. Distribution of data across five folds

To compare our tabled models with KT and PFA, slight modifications were made to the standard modeling procedures. Unlike tabling, these models carry the benefit of being able to predict performance on a student's first opportunity within a skill. Based on a 'prior knowledge' parameter, KT is able to predict the student's initial knowledge state, K_1, and therefore their performance on the first question, Q_1. Similarly, the equation for PFA defaults a prediction of the skill's difficulty parameter, β, as the student's initial state. These values essentially define a baseline for the student's knowledge, prior to any practice. Thus, in order to provide a fair comparison to tabled models, these first opportunity predictions were removed by shifting predictions to align with our 'next problem' analysis. Within KT, all subsequent skill opportunities were predicted using Expectation Maximization, a standard method for parameter learning within KT. The model was supplied the following initial parameters as 'ground truths' to begin the hill climbing process: *prior knowledge* = 0.30, *learn rate* = 0.20, *forget rate* = 0.00, *guess rate* = 0.20, and *slip rate* = 0.08. Within PFA, all subsequent skill opportunities were predicted by updating the equation presented in the previous section. These modifications resulted in the same number of data points for each model, providing grounds for fair comparison of the models.

Further, as noted briefly in the Dataset section, all models were restricted to 15 predicted opportunities per student per skill. This method was chosen largely to reduce the computation time required to fit KT using five-fold cross validation on such an extensive dataset. By capping the opportunity count, analysis time was reduced to approximately 20 hours. Other models were far less time intensive, all taking under three hours to arrive at predictions. Setting this restriction also served to reduce potential skewing in student level analyses by removing outliers with extensive opportunity counts.

RESULTS

All models were compared using the fit statistics of RMSE, R^2, AUC, and model accuracy. As the tabled models were not restricted to binary input, fit statistics were also found for consideration of modeling partial credit next problem correctness.

For each model, these statistics were found at the problem log level, the skill level, and the student level where merited. These statistics were then averaged across the level of analysis, resulting in the findings presented in Table 6, Table 7, and Table 8, respectively. Thus, at the problem log level, fit statistics were determined overall for the 288,307 predictions, without consideration of student or skill before being averaged across all problems. At the skill level, ten sets of fit statistics were determined (one set for each skill), which were then averaged across skills. At the student level, 15,253 sets of fit statistics were determined (one set for each student), which were then averaged across students. The latter two procedures were intended to

Model	Binary NPC				Partial NPC			
	RMSE	R^2	*AUC*	*Accuracy*	*RMSE*	R^2	*AUC*	*Accuracy*
Partial Credit + Problem Difficulty	0.4241	0.0674	0.6365	0.7310	0.3326	0.1062	0.5395	0.7298
Partial Credit	0.4244	0.0660	0.6309	0.7309	0.3327	0.1060	0.5351	0.7298
Problem Difficulty	0.4379	0.0057	0.5464	0.7300	0.3511	0.0043	0.3953	0.7298
Knowledge Tracing	0.4240	0.0680	0.6621	0.7298	--	--	--	--
Performance Factors Analysis	0.4227	0.0738	0.6644	0.7485	--	--	--	--

Table 6. Problem Level Average RMSE, R^2, AUC, and Accuracy for Models Predicting Next Problem Correctness (NPC)

Model	Binary NPC				Partial NPC			
	RMSE	R^2	*AUC*	*Accuracy*	*RMSE*	R^2	*AUC*	*Accuracy*
Partial Credit + Problem Difficulty	0.4224	0.0670	0.6300	0.7414	0.3284	0.1032	0.5130	0.7399
Partial Credit	0.4229	0.0656	0.6290	0.7414	0.3284	0.1031	0.5103	0.7399
Problem Difficulty	0.4364	0.0046	0.5323	0.7402	0.3473	0.0037	0.3560	0.7399
Knowledge Tracing	0.4225	0.0602	0.6500	0.7466	--	--	--	--
Performance Factors Analysis	0.4212	0.0664	0.6506	0.7499	--	--	--	--

Table 7. Skill Level Average RMSE, R^2, AUC, and Accuracy for Models Predicting Next Problem Correctness (NPC)

Model	Binary NPC				Partial NPC			
	*RMSE**	R^2	*AUC*	*Accuracy*	*RMSE**	R^2	*AUC*	*Accuracy*
Partial Credit + Problem Difficulty	0.3864	0.1027	0.5431	0.7684	0.2702	0.1108	0.3593	0.7674
Partial Credit	0.3866	0.0994	0.5392	0.7683	0.2701	0.1057	0.3619	0.7674
Problem Difficulty	0.4064	0.0829	0.5219	0.7676	0.2941	0.0851	0.3145	0.7674
Knowledge Tracing	0.3897	0.1057	0.4425	0.7729	--	--	--	--
Performance Factors Analysis	0.3882	0.0970	0.5003	0.7754	--	--	--	--

*R^2, AUC, and Accuracy are reported with less data than RMSE due to the nature of student level data.

Table 8. Student Level Average RMSE, R^2, AUC, and Accuracy for Models Predicting Next Problem Correctness (NPC)

properly weight skill and students based on their contribution to the dataset, thereby improving measures of model fit.

Student level statistics of RMSE were calculated based on all predictions. However, it should be noted that measures of R^2, AUC, and model accuracy could not be calculated for students with less than three skill opportunities. This discrepancy should affect all models equally, and thus we provide these measures for comparison in Table 8 with the caveat that they should not be directly compared to measures of student level RMSE.

DISCUSSION

The fit statistics for both the problem log and skill level generalizations paint very similar pictures of the relative success of our tabling method. The combined Partial Credit and Problem Difficulty model performs about as well as KT at both levels of analysis. At these levels, PFA appears to be the 'best' model for predicting binary next problem correctness, showing the lowest RMSE and highest AUC and model accuracy. However, we feel that a simple tabling method that can be performed with extreme efficiency yet still meets the standards of KT is well worth discussion.

Our first research question, "Does an algorithmically determined partial credit score outperform binary metrics when used to predict next problem correctness?" was answered with mixed results for binary predictions. Considering problem log level analysis, while KT and PFA attained fit statistics relative to those accepted in the field, our tabling method for partial credit considered alone only slightly underperformed these standards (RMSE = 0.4244, R^2 = 0.0660, AUC = 0.6309, Accuracy = 0.7309). However, when considering student level analysis, our partial credit tabling method outperformed both KT and PFA in terms of RMSE and AUC. To confirm that these findings were significantly different, we used a two-tailed paired samples t-test for RMSE comparison at both the student level and skill level. RMSEs obtained using our

tabling method with partial credit alone were significantly different from those found using KT at the student level, $t = 5.65$, $p < .001$, but were not significantly different at the skill level, $t = -1.65$, $p = 0.133$. Thus, it is difficult to tell if this finding is truly significant.

Our second research question, "Does a binned metric of current problem difficulty (e.g., Low, Medium, or High difficulty) provide a valid prediction of next problem correctness?" was answered by assessing the "Problem Difficulty" model. When taken alone, problem difficulty is not very helpful in predicting next problem correctness. This was the worst performing model across all granularities of analysis. A paired samples t-test was again used to compare student level and skill level RMSEs to those observed using the KT model. RMSEs obtained using our tabling method for Problem Difficulty were significantly worse than those found using KT at the student level, $t = -41.27$, $p < .001$, as well as those found using KT at the skill level, $t = -9.93$, $p < .001$. Of the tabled models, this model was also the lowest performing model when considering predictions of partial next credit correctness, drastically underperforming models that considered current problem partial credit score. Thus, we argue that problem difficulty alone is a poor metric for modeling student performance.

Our final research question, "Can current problem difficulty supplement partial credit score to outperform similar modeling techniques?" was answered by assessing the fit statistics for the combined "Partial Credit + Problem Difficulty" model. At the student level, this model outperformed both KT and PFA on predictions of binary next problem correctness as measured by RMSE (0.3864) and AUC (0.5431). This finding was significant using a two-tailed paired samples t-test comparing student level RMSEs, $t = 6.50$, $p < .001$, but was not significant when considering skill level RMSEs, $t = -1.34$, $p = 0.214$. Despite the low performance of the Problem Difficulty model, this combined model consistently outperformed partial credit when modeled alone, suggesting possible mediation effects. Using a paired t-test comparison, this difference was significant at the student level, $t = -4.55$, $p < .001$, but was not significantly reliable at the skill level, $t = -1.03$, $p = .310$. As such, it is difficult to quantify the potentially negative impact of considering problem difficulty when using partial credit to model next problem correctness.

Model fit indices for the prediction of partial credit scores for next problem correctness are provided for further consideration, but do not specifically link to our research questions. Drastic improvements in model fit suggest that intelligent tutoring systems should incorporate partial credit scoring as it has the potential to enhance the precision of student modeling. In the current study, these findings cannot be compared to standard KT and PFA models that utilize binary input and essentially predict binary performance. Future research will incorporate modifying these models to predict continuous partial credit metrics, thus allowing for further comparison.

CONTRIBUTION

The results from the present study suggest that considering partial credit for each skill opportunity can enhance the accuracy of student modeling. While the concept of using a tabling method to establish partial credit metrics that predict binary correctness is not novel [12], tabling a model based on algorithmically determined partial credit is, to the best of our awareness, a unique approach. This method was shown to perform about as well as KT when predicting binary next problem correctness. We feel that this finding still provides a significant contribution to the field, as KT is far more computationally expensive. Our KT analysis took approximately 20 hours to run, while all tabling methods were conducted by hand in less than three hours. While this is impressive in and of itself, the second author was then able to implement the tabling method presented here within the ASSISTments test database, arriving at a replication of our predictions in less than two minutes. If automated in such a manner, our Partial Credit + Problem Difficulty model could predict next problem performance on par with KT in approximately one 600^{th} of the time. This increase in efficiency could prove essential for intelligent tutoring systems that currently incorporate KT models to adaptively control student skill practice.

Further, the partial credit model was novel in its ability to predict partial credit scores for next problem correctness, thereby enhancing model fit even further. In future research, we hope to modify the standard KT and PFA models to allow for the prediction of continuous variables for comparison. We also anticipate directly comparing our partial credit model to the "Assistance" Model established in previous research [12]. The "Assistance" Model cited a clear cut, albeit subjective, method for the provision of partial credit scores. As the tabling technique employed made predictions on a continuous scale rather than by binning partial credit as we have shown in the present study, comparison was not presently possible without ensembling our findings with standard KT measures [12]. However, as alternating ensembling techniques lead to inconsistent results [4], we argue for direct modifications within KT that will allow the model to learn partial credit scores at each opportunity and to gauge a student's knowledge state on a continuum. A similar model was previously suggested by Pardos & Heffernan [10], but to our knowledge has never been implemented. Thus, the present study lays the groundwork for future research in modifying KT.

The assessment of models considering problem difficulty also provides a contribution to the modeling literature. It seems intuitive that problem difficulty should influence a students' ability to answer the current problem correctly, and that it likely influences their knowledge state and next problem correctness. The findings here suggest

that problem difficulty alone, when binned into generic groups of Low, Medium, and High difficulty, does not provide accurate models of next problem correctness. However, problem difficulty appeared to enhance modeling when coupled with partial credit in comparison to partial credit modeled alone, although this difference was not shown to be significant. Still, we believe that some measure of problem difficulty is important to consider when modeling student learning. Future research should investigate using a continuous metric or designing an alternative binning approach for this feature. Future work should also consider devising an approach to remedy the issue of being unable to predict a student's first opportunity within a skill when using tabled models. Possible solutions include per student estimates of prior knowledge based on performance on other skills within the tutor, or simply implementing problem difficulty as a measure of likelihood for accuracy.

Despite the impressive performance of our partial credit model, we retain skepticism in regards to the subjective nature of our partial credit algorithm. As multiple arbitrary partial credit models have now been designed to assess log data from the ASSISTments platform [12], we argue for the design of a data driven algorithm that considers and compares a myriad of logged features. Future work will examine a grid search of possible hint and attempt penalties to examine the sensitivity of the approach described herein. The data files of intelligent tutoring system are rich with information pertaining to students' actions, including the time required for first response, their sequence of actions within each problem, and the specific misconceptions that are driving incorrect responses. These features may provide critical information for the scoring of partial credit. When considering the approach used in the present study, using an algorithm to establish partial credit scores prior to tabling provides the leeway for tabled models to consider these additional features. Future research could easily replicate similar models, combining partial credit with novel features for additional exploration of the observed effect.

ACKNOWLEDGMENTS

We acknowledge funding from NSF (grant #'s 1316736, 1252297, 1109483, 1031398, 0742503), ONR's "STEM Grand Challenges," and IES (grant #'s R305A120125, R305C100024). Thanks to S.O. & L.P.B.O.

REFERENCES

1. Attali, Y. & Powers, D. (2010). Immediate feedback and opportunity to revise answers to open-end questions. *Educational and Psychological Measures*, 70 (1), 22-35.
2. Corbett, A.T., Anderson, J.R. (1995). Knowledge Tracing: Modeling the Acquisition of Procedural Knowledge. *User Modeling and User-Adapted Interaction*, 4: 253-278.
3. Duong, H.D., Zhu, L., Wang, Y., & Heffernan, N.T. (2013). A Prediction Model Uses the Sequence of Attempts and Hints to Better Predict Knowledge: Better to Attempt the Problem First, Rather Than Ask for a Hint. In S. D'Mello, R. Calvo, & A. Olney (Eds.) Proceedings of the 6th International Conference on Educational Data Mining. Memphis, TN. 316-317.
4. Gowda, S.M., Baker, R.S.J.D, Pardos, Z., & Heffernan, N.T. (2011). The Sum is Greater than the Parts: Ensembling Student Knowledge Models in ASSISTments. Proceedings of the KDD 2011 Workshop on KDD in Educational Data.
5. Koedinger, K.R. & Corbett, A.T. (2006). Cognitive tutors: Technology bringing learning science to the classroom. In K. Sawyer (Ed.), *The Cambridge handbook of the learning sciences* (61-78). New York: Cambridge University Press.
6. Murphy, K. (2001). The Bayes Net Toolbox for MATLAB. Computing Science and Statistics, 33(2), 1024-1034.
7. Author1. (2014). L@S 2015 Submission: Dataset. Retrieved 10/14/14, http://tiny.cc/LaS2015Submission
8. Pardos, Z.A. & Heffernan, N.T. (2010). Modeling Individualization in a Bayesian Networks Implementation of Knowledge Tracing. In Proceedings of the 18th International Conference on User Modeling, Adaptation and Personalization. 255-266.
9. Pardos, Z.A., & Heffernan, N.T. (2011). KT-IDEM: Introducing Item Difficulty to the Knowledge Tracing Model. In Joseph A. Konstan et al. (Eds.): UMAP 2011, LNCS 6787, 243-254.
10. Pardos, Z.A. & Heffernan, N.T. (2012). Tutor Modeling vs. Student Modeling. Proceedings of the Twenty-Fifth International Florida Artificial Intelligence Research Society Conference, 420-425.
11. Pavlik, P.I., Cen, H., Koedinger, K.R. (2009). Performance Factors Analysis - A New Alternative to Knowledge Tracing. In: Proceedings of the 14th International Conference on Artificial Intelligence in Education, Brighton, UK, 531-538.
12. Wang, Y. & Heffernan, N.T. (2011). The "Assistance" Model: Leveraging How Many Hints and Attempts a Student Needs. The 24th International FLAIRS Conference.
13. Wang, Y. & Heffernan, N. (2013). Extending Knowledge Tracing to Allow Partial Credit: Using Continuous versus Binary Nodes. In K. Yacef et al. (Eds.) AIED 2013, LNAI 7926, 181-188.