THE BAYESIAN TRACING MODEL IN COURSE SOFTWARE

Presented to
The Faculty of the School of Engineering and Applied Science
In Partial Fulfillment of the Requirements for the Degree
Bachelor of Science in Computer Science

By

Carrington Murphy

April 28, 2020

On my honor as a University student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments.

ADVISOR
Mark Floryan, Department of Computer Science

THE BAYESIAN TRACING MODEL IN COURSE SOFTWARE

Abstract.

In order to provide additional feedback and incentivize mastery, a knowledge tracing algorithm was implemented as part of a course software. The algorithm calculates student grades based on performance on quizzes. This paper investigates the parameters in the algorithm to determine its effectiveness in modelling different situations. We conclude that the model will effectively model learning in this context.

INTRODUCTION

In the Computer Science department at the University of Virginia, there is a core series of classes that is part of the curriculum for all students studying computer science. This study will examine one of these courses, CS 2150, Program and Data Representation, which is a pre-requisite for most of the upper level electives. The material covered in this class builds many of the fundamentals for success in later courses. As a result, it is imperative that students are comfortable with the concepts taught in this course.

Mark Floryan, a professor at the University of Virginia, believes that many students may be completing the course, and other similar courses, without solid understanding of all of the content. His hypothesis is based on multiple semesters of experience teaching this course along with upper level electives. According to his research, about 40 percent of the class has a grade difference of at least 20 percent between their homework average and exam average in CS 2150. In order to remedy the problems with this course, Floryan has proposed the adoption of a course software focusing on mastery. The project also aims to provide students with specific grading guidelines and live feedback. It is hypothesized that by using this system students may have better understanding of what is expected of them and their current standing in the class. This understanding may support greater student confidence and retention of material.

The redesigned course will be based in a software solution called the Student Performance Tracker (SPT) designed and implemented by Floryan and a team of computer science students. The entire course is modelled as a directed acyclic graph, which depicts both the progression of material in the class and the flow of knowledge from topic to topic. This graph is shown in Figure 1.

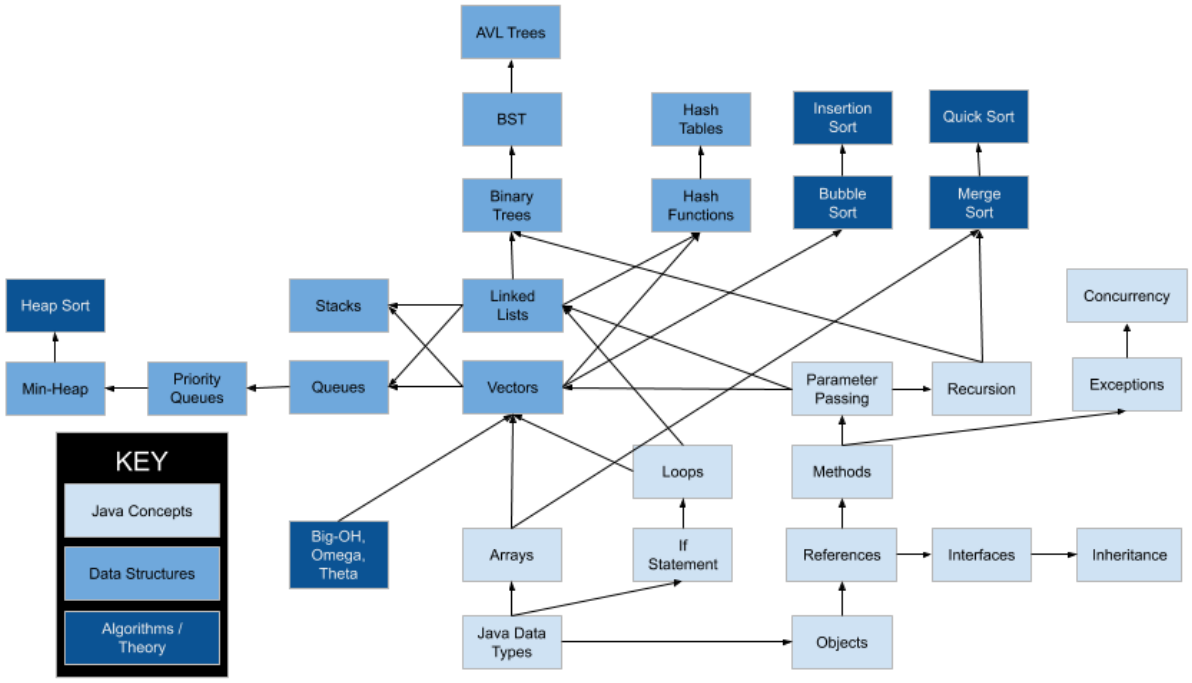

Figure 1: Course Graph for a data structures course taught at UVa.

Each node in the graph is a topic which will be taught over a short period of time. In order to calculate students' grades, each node will have a score assigned to it. Over time, students will have opportunities to try to raise that score. Once the student's score is above a set threshold, the student will receive a passing grade on that node. At the end of the class, the overall grade received in the course will be a direct result of the number of topics passed. This should reflect the percentage of the material in the class that has actually been mastered. Hopefully, this new system will encourage mastery of all concepts in the course. This project is focused on the design and development of a model for computing a student's score at each individual node as part of the SPT software.

Although there have been many proposed ways of modelling learning throughout the years, there are few more used than the 1995 Bayesian Knowledge Tracing (BKT) model (Corbett & Anderson, 1995). It has been widely employed in both online tutoring systems and analytics including Carnegie Learning and ASSISTments (Feng, Heffernan, Ruiz & Beck, 2006; Ritter, Anderson, Koedinger & Corbett, 2007). Though the model is designed for measuring learning of discrete skills based on a series of tasks, it can be widely applied in a variety of situations. Much research has been done on extensions and modifications to the algorithm which

can attempt to better predict student learning. However, because of its simplicity and easy to understand nature, the classic implementation of BKT provides an ideal solution to calculate individual scores for each node in. The model can be easily explained to students, provides live feedback and updates on students' scores in a topic, and requires minimal resource use in terms of time and storage. Additionally, the model is highly adaptable and can be fit to a variety of situations.

The BKT model as originally introduced by Corbett and Anderson is a function of four parameters. Each parameter affects the model behavior separately. Those parameters are as follows:

- *P(S)* is the probability of making a slip, if student does know the skill.
- *P(G)* is the probability of guessing correctly, if the student does not know the skill.
- *P(T)* is the probability of learning the skill, if the student does not know the skill.
- $P(L_0)$ is the probability that the student initially knows a particular skill.

The parameters are used in two equations which are commonly called the Knowledge Tracing Algorithm. The algorithm functions by updating the $P(L)$ value based on whether the student correctly or incorrectly performs the desired action. The equations are as follows:

$$P(L_i|correct) = 1 - \frac{P(T) \cdot \big(1 - P(L_{i-1})\big) \cdot P(G)}{P(G) + \big(1 - P(S) - p(G)\big) \cdot P(L_{i-1})}$$

$$P(L_i|incorrect) = 1 - \frac{P(T) \cdot \big(1 - P(L_{i-1})\big) \cdot (1 - P(G))}{1 - P(G) - \big(1 - P(S) - P(G)\big) \cdot P(L_{i-1})}$$

These equations are shown below in Figure 2. The model can be reasonably understood as a student's predicted understanding given a response, and the model parameters (where L_0 becomes L_i, the student's score after attempt *i*). Given a correct or incorrect response, the score will update according to the green or red curve, respectively.

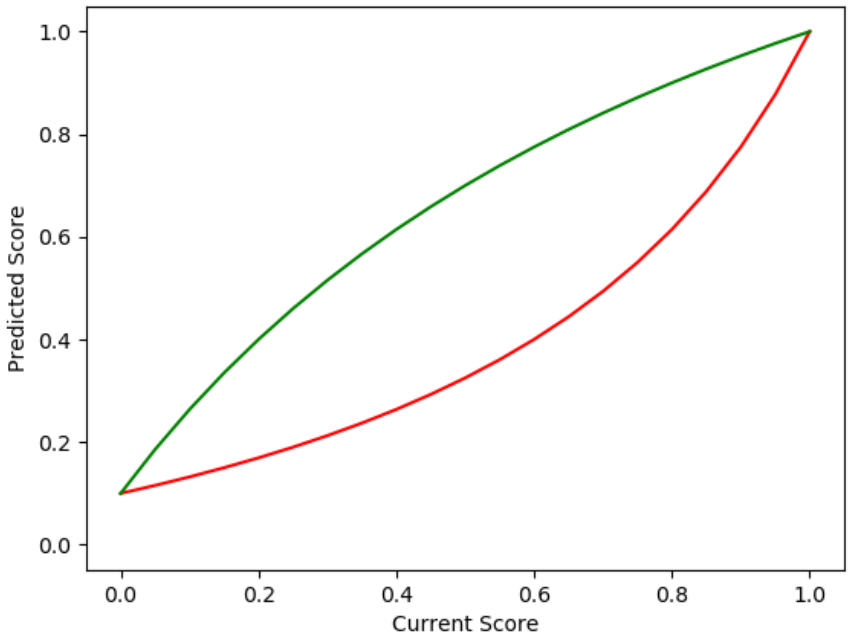

Figure 2: Bayesian Tracing Model – Knowledge Tracing Algorithm – functional form. Model parameters are: P(L) = 0.3, P(G) = 0.4, P(S) = 0.2, P(T) = 0.1

Many studies have focused on fitting the parameters of a BKT model using data. The data is then used to make a prediction about future student performance. In this application, the BKT model is instead used to grade the student's performance and to provide live feedback. In order to best understand this adaptation, it is perhaps best to provide a framework for understanding and applying BKT models to this purpose. For this reason, this study will provide an overview of the parameter space for the BKT algorithm and provide recommendations for teachers to use this design in grading. Through this work, we hope to present a useful summary of the ways in which differences in model parameters effect the predicted scores for different students. In this way, teachers can use the designs discussed to fit the model, not only to student data, but to their own grading preferences.

The goals for this project are as follows:

1. To implement the BKT algorithm for modelling learning as part of the SPT software.
2. To investigate and analyze the parameter space for models used in the context of college computer science classes in order to maximize effectiveness.

METHODS

In order to discover appropriate ranges for the parameters, a simple implementation of the BKT was created for preliminary investigation. Because the back-end of the larger software used Python and Django, it was decided that the model should also be built in Python so that it could be easily added into the existing code base. At first, a class-based model was developed, which stored the model as an instance of the BKT class. However, as communication with Floryan and the development team continued, it became clear that in the current architecture, we would only need one instance of the model, and that it would be helpful to implement a functional solution which could be called from the front-end. Both solutions used the equations described previously to calculate an update to the student's grade based on their response.

In pursuit of the second goal of this project, the implementation needed to be able to run simulated data through the model. Using the pre-built class-based solution, a simulation was set up which would be able to run a series of student data through the model and calculate the scores at each step for each student. Once it was supplemented with a graphing library the whole software allowed for visualization of many sample students over time.

In order to effectively summarize the effects of differences in the model parameters, it is important to understand the ways in which the parameters actually effect the shape of the model, in addition to the ways in which the model effects students' scores. The first part of the procedure involved graphing a variety of different models to view their properties which might be relevant for teachers using the model. Among those properties are: minimum achievable score, slope (rate of score changes), and the shape of correct and incorrect curves.

In addition to the different models, a variety of sample response data is analyzed. It would be relevant to understand not only how the model functions, but how differences in the model parameters can change the predicted score given the same string of student responses. The projected students were generated by estimating possible student patterns. Those patterns are displayed in Table 1 below. Student A rarely makes a mistake, Student B has little to no understanding, Student C is inconsistent, and Student D improves over time. The colors refer to the student graphs in the results section.

Student	Response Pattern (1=correct, 0=incorrect)	Color
A	1, 1, 1, 1, 1, 1, 1, 1, 1, 1	Green
B	0, 0, 0, 0, 0, 0, 0, 0, 0, 0	Red
C	1, 0, 1, 0, 1, 0, 1, 0, 1, 0	Orange
D	0, 0, 0, 1, 0, 1, 0, 1, 1, 1	Blue

Table 1: Students and Response Patterns.

Given the mathematical properties and the sample students, it then becomes pertinent to connect them to the more high-level grading philosophy inherent in the model. For example, for a model with a flatter curve for correct responses than incorrect ones, how does it affect the negative and positive incentives that will be portrayed through the updating of scores. Generally, the final section will seek to provide conclusions about how to observe the model and how the models might better fit a certain style of teaching or a philosophy about grading. Initially, when setting the parameters, the instructor may have a rough idea of the effects of their choices, however they may be able to make better decisions based on observations of students over time (or otherwise changes in grading philosophy). This section is meant to provide instructors with the ability to make those observations and changes.

RESULTS

The first parameter, L_0, does not directly affect the shape of the model, simply where the student begins on the spectrum. L_0 may have more significant effects on the later parts of this study, but it has minimal effects on the model itself.

The second parameter, G, does affect the shape of the model, specifically, it affects the shapes of the correct and incorrect curves. Figure 3 shows three models using different values for G.

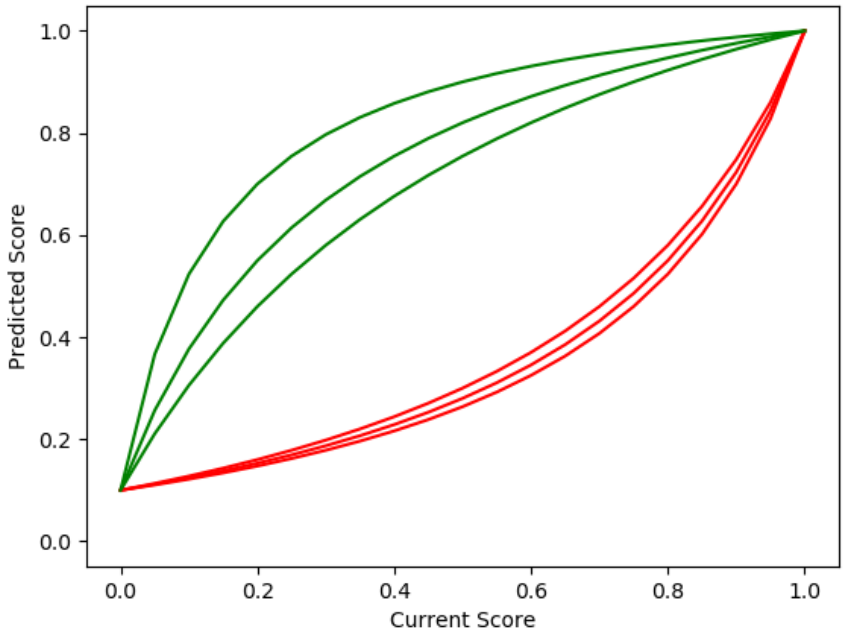

Figure 3: Bayesian Tracing Model – Knowledge Tracing Algorithm – functional form. Model parameters are: P(L) = 0.3, P(S) = 0.2, P(T) = 0.1

As the value of G increases, the shapes of both curves flatten. The shape of the correct curve, shown in green, changes more drastically with incremental changes in G. The graph's overall slope changes minimally.

The third parameter, S, affects the shape of the model, specifically, it affects the shapes of the correct and incorrect curves. Figure 4 shows a series of models using different values for S.

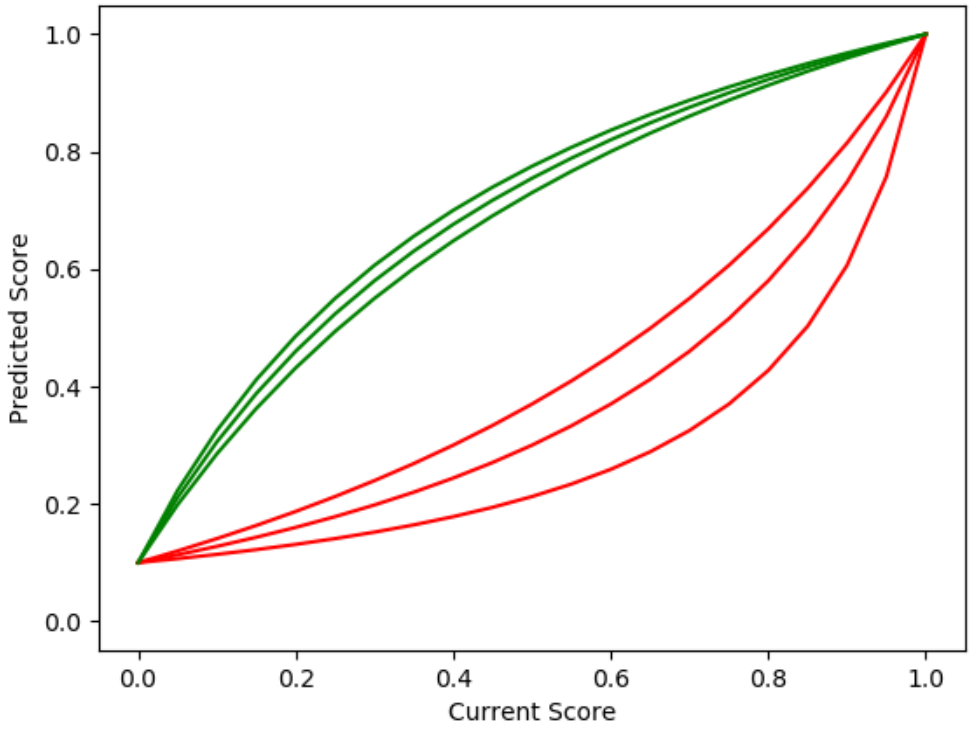

Figure 4: Bayesian Tracing Model – Knowledge Tracing Algorithm – functional form. Model parameters are: P(L) = 0.3, P(G) = 0.3, P(T) = 0.1

As the value of S increases, the shapes of both curves flatten. The shape of the incorrect curve, shown in red, changes more drastically with incremental changes in S. The graph's overall slope changes minimally.

The fourth parameter, T, affects the slope of the entire graph. Figure 5 shows a series of models using different values for T.

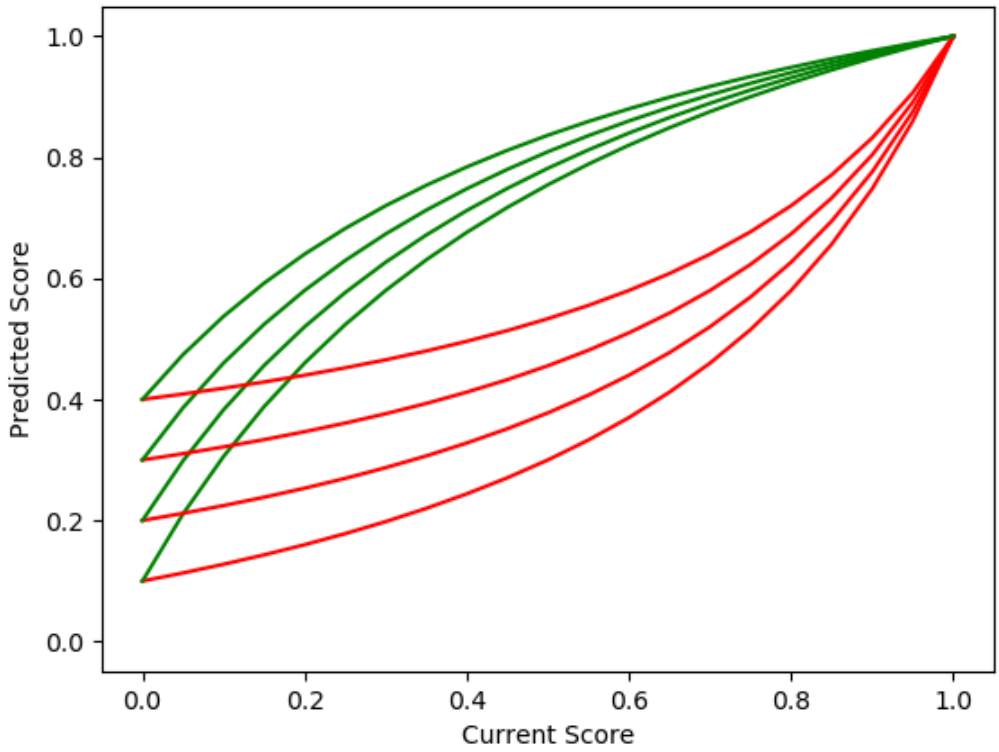

Figure 5: Bayesian Tracing Model – Knowledge Tracing Algorithm – functional form. Model parameters are: P(L) = 0.3, P(G) = 0.3, P(S) = 0.2

As the value of T increases, the slope of the entire graph decreases. The y-intercept of both curves is always exactly equal to the value of T. This indicates that the minimum predicted score is always at least T. The shapes of the curves change, but minimally with respect to the overall slope.

Analysis of Students

To further understand the importance of these models, it is important to understand the implications of the parameters on students. Other papers (Van de Sande, 2013) have already discussed the mathematical and theoretical limitations on these parameters, however, it is unclear how the parameters effect the scores of students over time given sample data.

The first parameter, L_0, as previously stated, does not play a role in determining the structure of the model. It simply designates the "starting point" where students' predicted scores are initialized. For the purposes of this discussion, it will be assumed that students have little to no previous knowledge of the subject. Another way to phrase this is that students must prove that

they understand a topic by starting at a low prediction. This assumption will be discussed further in the next section.

The second parameter, G, is analyzed in Figure 6 below. Each model has a different G parameter, and the student's data can be seen over the course of their attempts. All four models currently have a threshold set at 0.9. This number can be set arbitrarily, but will remain consistent through this analysis.

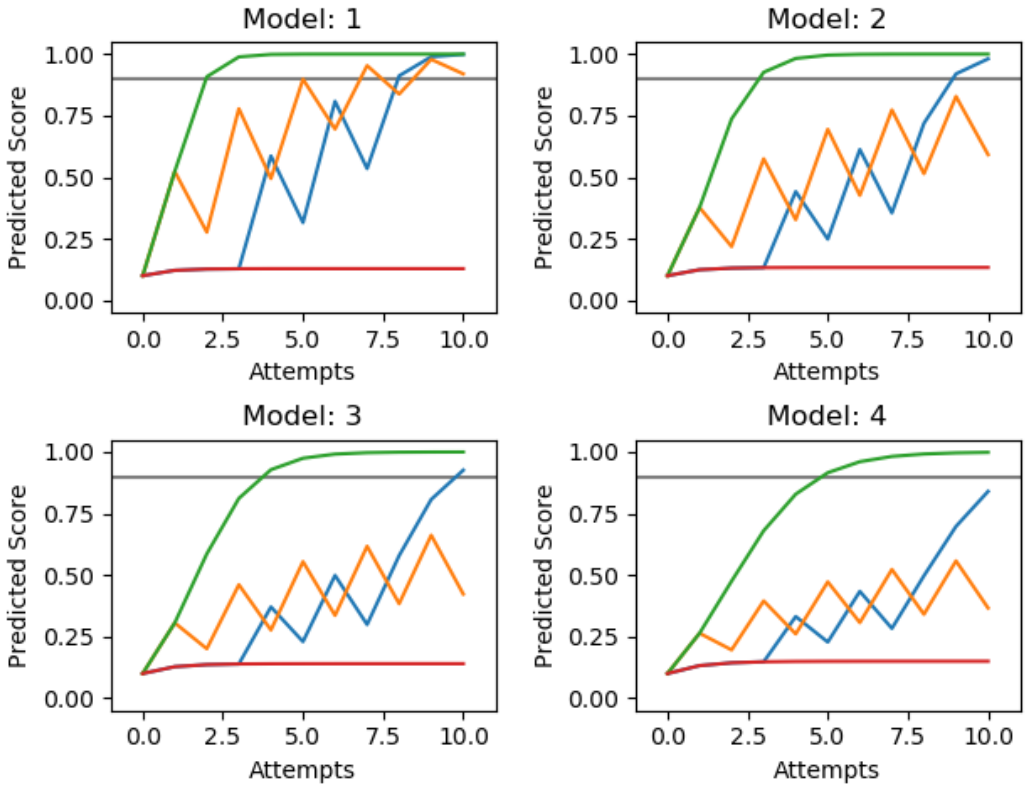

Figure 6: Student performance vs number of attempts.
Overall model parameters are: P(L) = 0.1, P(S) = 0.2, P(T) = 0.1.

Model 1: P(G) = 0.1
Model 2: P(G) = 0.2
Model 3: P(G) = 0.3
Model 4: P(G) = 0.4

One of the most significant factors distinguishing these graphs is the number of attempts required for student A to achieve mastery. Of the four, only Model 4 requires at least five correct responses in a row before awarding mastery to student A. Additionally, it is interesting to investigate student C, in orange, whose score increases despite alternating between correct and

incorrect responses. In models 1 and 2, student C seems to converge to a predicted score which is significantly higher than 50%. This may be reasonable in some applications, but it is important to note that setting G to a value in the range of 0.1 to 0.2 will occasionally exhibit this property. Student D, who starts off poorly and improves over time, achieves mastery significantly earlier when the G is 0.1 or 0.2. In Model 4, student D has a relatively low score for a longer period of time and only starts to raise that score when they show that success.

The third parameter, S, is analyzed in Figure 7 below. Each model has a different S value.

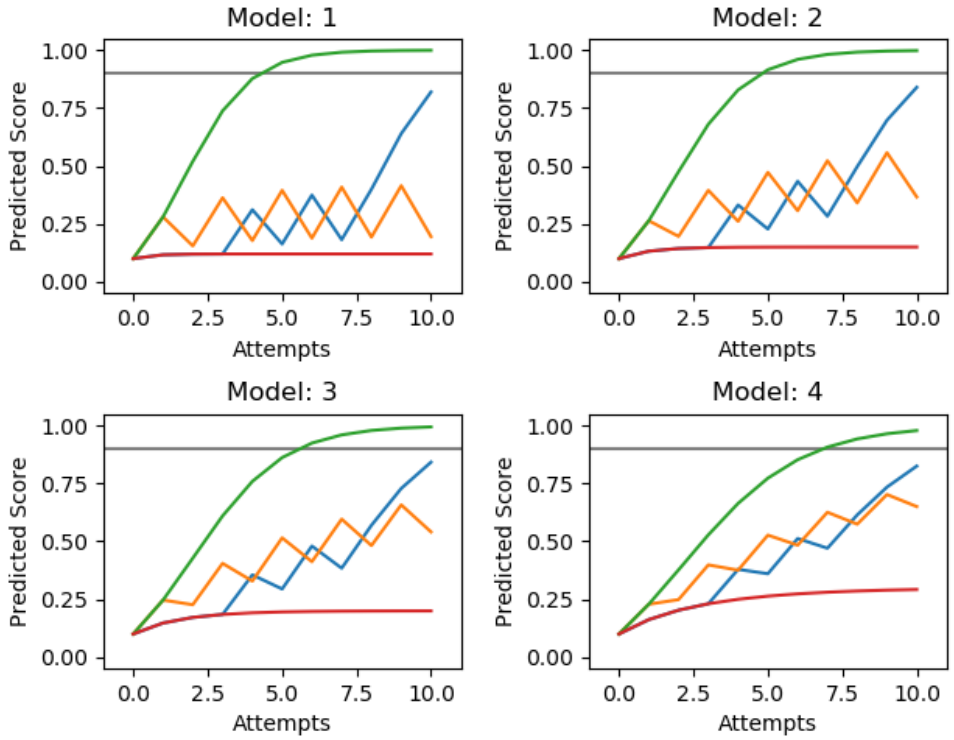

Figure 7: Student performance vs number of attempts.
Overall model parameters are: P(L) = 0.1, P(G) = 0.4,
P(T) = 0.1.

Model 1: P(S) = 0.1
Model 2: P(S) = 0.2
Model 3: P(S) = 0.3
Model 4: P(S) = 0.4

The slip parameter will significantly affect the consequences of incorrect answers. The slip parameter does not change whether any students pass the threshold. Student A still succeeds in around 5 or 6 correct attempts. Student C and D, who responded both correctly and incorrectly, are clearly less punished for their incorrect attempts. Model 4 in particular seems to exhibit some of the behavior previously mentioned where students who alternate correct and

incorrect responses actually increase their score to significantly higher than 50%. Higher slip parameters like that of Model 4 are much less punishing of incorrect answers, allowing students to retain much of their progress. Student B, who had no correct responses, actually sees his score increase slightly. It is most clear in Model 4. This can be explained by the fact that even if Student B answers incorrectly, it is possible that they did learn the skill, they simply made a mistake. Compounding multiple incorrect answers in a row does not actually demonstrate knowledge, but it can increase a student's score because it demonstrates that they have had a number of opportunities to learn the skill, and they may have learned it recently and made a small mistake. This property will be discussed further in the next section.

The fourth parameter, T, is analyzed in Figure 8 below.

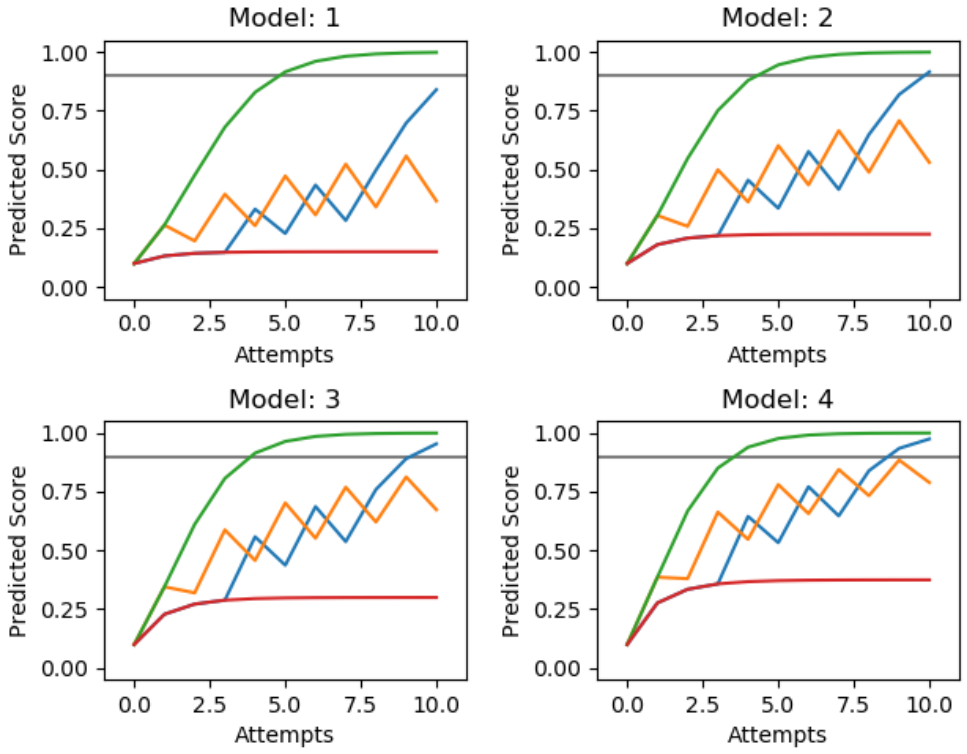

Figure 8: Student performance vs number of attempts. Overall model parameters are: P(L) = 0.1, P(G) = 0.4, P(S) = 0.2.

Model 1: P(T) = 0.1
Model 2: P(T) = 0.15
Model 3: P(T) = 0.2
Model 4: P(T) = 0.25

The transfer parameter, T, plays a significant role in the shape of the model and the student scores. Student A achieves mastery much faster with a higher value of T. With $P(T) = 0.25$,

student A is almost able to achieve mastery in only three attempts, a similar effect to that seen from having a guess rate of 0.1. The increasing alternating response score (seen in student C and D) is also in effect once T goes much above 0.15. Overall, increasing T greatly benefits students scores. Students are punished less for their mistakes, and are generally rewarded for attempting problems, because the model will take into account the chance that they may have learned the task on that attempt.

DISCUSSION

Based on this set of observations, T s one of the most important parameters to align with the teaching philosophy. A higher value of T assumes that more learning is taking place during the quiz. It gives students a generous boost to their score just for attempting a problem. Because of this increase, it will often be easier to achieve mastery. In order to retain that difficulty, the G parameter may be increased artificially. Keeping T low is more aligned with the idea of demonstrating mastery through consistency over time. This will require students to get a larger number of questions correct over a longer period of time. This philosophy should not be confused with the idea of making the model more punishing. Requiring consistency can be independent, especially if you allow students to make mistakes more freely by increasing the S parameter.

The G and S parameters are more aligned with the philosophies of positive and negative consequences. The G parameter, generally, describes how much reward the student will receive for correctly answering a question. Likewise, the S parameter describes how punishing the model will be for incorrect answers. For specific skills, it would make sense to have more significant positive and negative consequences. This follows based on the assumption that each individual question will reveal significantly more about a specific skill than about a general skill. In order to increase the consequences, G and S should be reduced. For more general topics, it would be more effective to have lower consequences for each answer. Because each individual question provides less information about a student's proficiency, this will require students to answer more questions correctly in order to progress. In order to decrease the consequences, G and S should be increased. These parameters should also be adjusted if questions are exceptionally difficult or simple. Simple questions should have a lower S parameter and a higher G parameter, because it

is unlikely a student misses the question. Difficult questions should have the inverse, higher S and lower G, because it is more likely the student misses the question.

The L_0 parameter, as previously discussed, is relatively ineffective at changing the model itself. Changes to this parameter are only reasonable if the model has some reason to assume that students already have knowledge of the topic. Based on this understanding, it is generally recommended to keep the L_0 parameter low.

Based on the findings of this research, the recommended starting model for the SPT software should require demonstrated consistency over time in general topic areas. The starting parameters chosen for this application are:

$$L_0 = 0.1$$
$$G = 0.4$$
$$S = 0.2$$
$$T = 0.1$$

Other studies have researched the effects of granularity on the effectiveness of the model. For example, Feng, Heffernan, Mani, and Heffernan (2006) found that a finer grain 78-skill yielded significantly better prediction results than the courser grain 5-skill model. Overall, this fits with the original description of BKT, which is meant for individual skills, called rules (Corbett & Anderson, 1995). However, because of the application of this software system for learning and quizzing students, rather than producing reliable prediction results, it is reasonable to suspect that BKT will be an effective model. For example, requiring students to answer more questions than the minimum required for mastery might actually be desirable in this application, because student will have additional practice with that skill. Also as admitted by Feng et al (2006), it is unknown how fine the granularity should be in order to maximize prediction results. Based on my analysis, the model is certainly flexible enough to function well for more general topics. It will likely be most effective when individual skills are modeled separately.

RECOMMENDATIONS FOR FUTURE WORK

Alongside this investigation, a number of additional research projects will be relevant. One major study should involve mapping and fitting student data from this system to the models to understand how accurate the selected parameters are. Additionally, there is more research to be done on the models themselves. One project could investigate the implications of different threshold cutoffs for a passing grade. Lastly, there have been many additional modifications to the BKT over the years. It would certainly be prudent to analyze the effectiveness of these other adaptations in this application. This includes research on topics such as partial transfer of skills, forgetting as a function of time, additional learning states, and others.

REFERENCES

Corbett, A. T., & Anderson, J. R. (1995). Knowledge tracing: Modeling the acquisition of procedural knowledge. User Modeling and User-Adapted Interaction, 4(4), 253-278.

Feng, M., Heffernan, N. T., Mani, M., & Heffernan, C. (2006). Using mixed-effects modeling to compare different grain-sized models. Educational Data Mining: Papers from the AAAI Workshop, 57-66.

Qiu, Y., Qi, Y., Lu, H., Pardos, Z. A., & Heffernan, N. T. (2011). Does time matter? Modeling the effect of time with Bayesian knowledge tracing. EDM 2011 - Proceedings of the 4th International Conference on Educational Data Mining. 139-148.

Ritter, S., Anderson, J., Koedinger, K., & Corbett, A. (2007). Cognitive tutor: Applied research in mathematics education. Psychonomic Bulletin & Review, 14(2), 249-255.

van de Sande, B. (2013). Properties of the Bayesian knowledge tracing model. Journal of Educational Data Mining, 5(2), 1-10.

Yudelson, M. V., Koedinger, K., & Gordon, G. J. (2013). Individualized Bayesian knowledge tracing models. Artificial Intelligence in Education. AIED 2013, 171-180.