Student Model Adjustment Through Random-Restart Hill Climbing

Ahmad Salim Doost
Saarland University
66123 Saarbruecken, Germany

Erica Melis
German Research Center for Articial Intelligence (DFKI)
Stuhlsatzenhausweg 3, 66123 Saarbruecken, Germany

Abstract

ACTIVEMATH is a web-based intelligent tutoring system (ITS) for studying mathematics. Its course generator, which assembles content to personalized books, strongly depends on the underlying student model. Therefore, a student model is important to make an ITS adaptive. The more accurate it is, the better could be the adaptation. Here we present which parameters can be optimized and how they can be optimized in an efficient and affordable manner. This methodology can be generalized beyond ACTIVEMATH's student model. We also present our results for the optimization based on two sets of log data. Our optimization method is based on random-restart hill climbing and it considerably improved the student model's accuracy.

1 Preliminaries

An intelligent tutoring system (ITS) is a computer system that provides personalized support (either by giving feedback or instruction) to students performing tasks without the intervention of human tutors. They can improve education quality by providing new possibilities (e.g. for homework assignments or self-study opportunities). Therefore, ITSs can be used to support teaching and diminish the teacher shortage problem [Flynt and Morton, 2009; Ingersoll and Perda, 2009] to some extent. ITSs can be permanently available, widely reusable, location-independent and adaptive. Adaptivity is an important feature of ITSs: it allows to focus the education to pedagogically useful content like teaching content the student has difficulties with and keeping the student from working on too easy or too difficult exercises. To make ITSs able to adapt to single students, an underlying student model aiming to represent certain aspects of a student (most importantly his/her skills), is required.

A student model represents variables and characteristics of a learner which are relevant for educational purposes. A typical learner variable is the student's state of knowledge concerning various concepts and competencies. This is continuously adjusted with the learner's progress over time. When developing a student model several parameters have to be determined which, in a first run, may be defined in an ad hoc way. However, these values are usually not appropriate and also wrong interrelations may cause problems. Instead, an evaluated configuration might improve the accuracy of the student model, which in turn leads to a better individualized learning experience.

How can we optimize parameters in a generic way? First of all, the parameters of ACTIVEMATH's student model, also called *Salient Learner Model* (SLM), which we want to optimize are described. We demonstrate how we adapted the Random-Restart Hill Climbing (RRHC) algorithm [Russel and Norvig, 2003] and used it to adjust SLM. Finally, we provide evaluation results and describe how the methodology can be generalized beyond SLM.

1.1 ACTIVEMATH and SLM

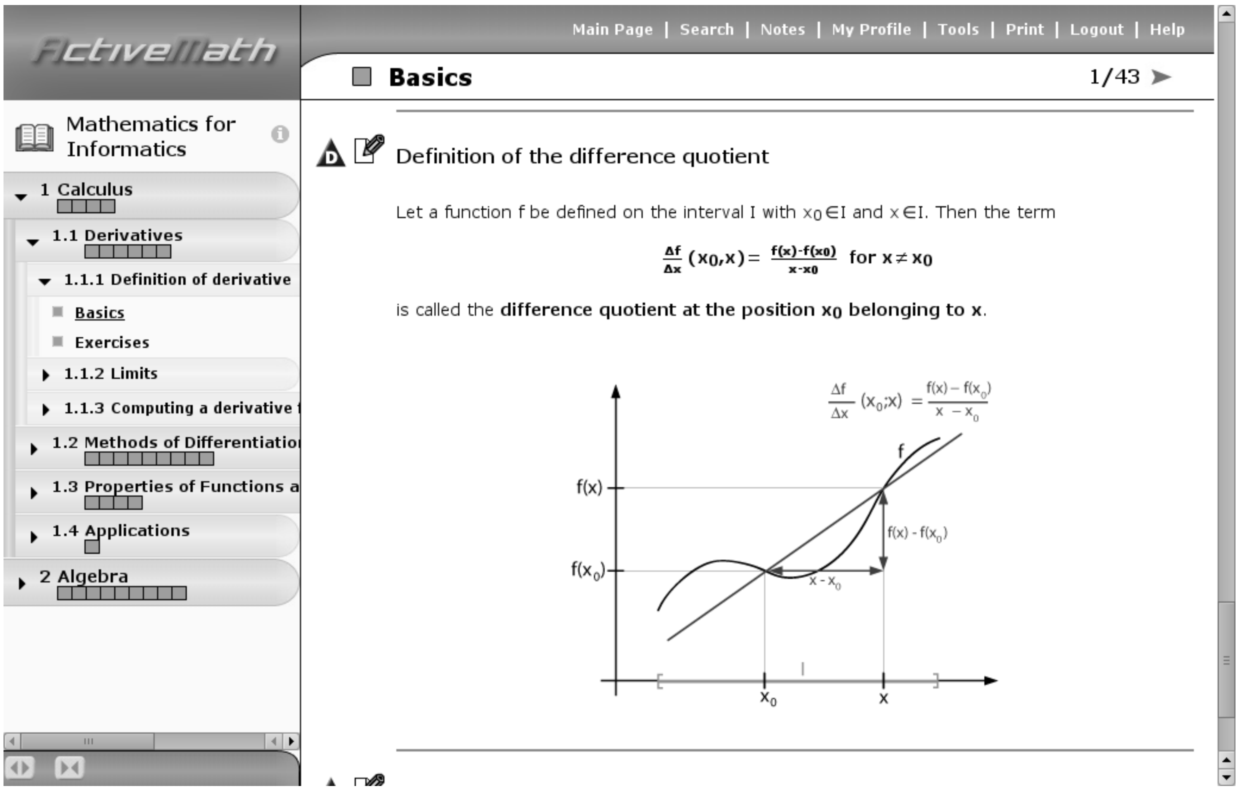
ACTIVEMATH is a web-based ITS for studying mathematics[1] [Ullrich and Libbrecht, 2008]. It is being developed at Saarland University and at the German Research Center of Artificial Intelligence (DFKI) since the year 2000. Its learning content (like definitions, examples, exercises and explanations), are semantically encoded in an extended OMDoc [Kohlhase, 2006]. The OMDoc documents are then converted by the presentation component to the desired output format (like HTML or PDF, cf. Figure 1).

Students cannot only browse through prerecorded static books. With the course generator of ACTIVEMATH, learning objects can be organized into a book which is specific to the student's abilities: The generated book depends on the student's goals and on the modeled belief about the learner's competencies. The goal can be for example to discover a new topic, revise an already known topic or simulate an exam. The content is selected according to special rules of the course generator and depends on the knowledge of the learner. The ACTIVEMATH course generator aims to generate user-specific courses with relevant learning material and appropriate difficulty. For example, if the student wants to discover a new topic but his competencies on some of the prerequisites for the topic are poor, then the course generator can include them in the personalized book with an easy level of difficulty. The content of the personalized dynamic books are updated and restructured with the student's progress over time.

The exercise system of ACTIVEMATH runs interactive problems with personalized feedback. Like other modules in ACTIVEMATH, the exercise system publishes events to notify the remaining parts of the system about the user interactions. The student model of ACTIVEMATH [Faulhaber and Melis, 2008] listens to these events and updates its belief after every exercise step. It takes the learner's achievements into account together with the exercise metadata (e.g. exercise difficulty).

The most relevant metadata is the set of trained concepts and cognitive processes. *Addition of fractions* or *Pythagorean theorem* are examples for concepts. Concepts

[1] `http://demo.activemath.org`


ActiveMath
Main Page | Search | Notes | My Profile | Tools | Print | Logout | Help
Basics
1/43
Mathematics for Informatics
1 Calculus
1.1 Derivatives
1.1.1 Definition of derivative
Basics
Exercises
1.1.2 Limits
1.1.3 Computing a derivative
1.2 Methods of Differentiatio
1.3 Properties of Functions a
1.4 Applications
2 Algebra
$\frac{\Delta f}{\Delta x}(x_0;x) = \frac{f(x) - f(x_0)}{x - x_0}$
f
f(x)
f(x) - f(x$_0$)
f(x$_0$)
x - x$_0$
I
x$_0$
x

Definition of the difference quotient

Let a function f be defined on the interval I with $x_0 \in I$ and $x \in I$. Then the term

$$\frac{\Delta f}{\Delta x}(x_0,x) = \frac{f(x)-f(x_0)}{x-x_0} \quad \text{for } x \neq x_0$$

is called the **difference quotient at the position x_0 belonging to x**.

Figure 1: A page of a book in ACTIVEMATH from a web-browser.

can be connected with each other with relations like the *prerequisite*-relation specifying that in order to be able to master one concept, the prerequisite-concept has to be mastered first (e.g. the concept *adding fractions with equal denominator* forms a prerequisite of the concept *adding fractions with unlike denominators*). The cognitive process metadata defines what process (like *detecting errors* or *applying algorithms*) is trained. We refer to the pair of a concept and a cognitive process as *competency*. For example, *detecting errors about the Pythagorean theorem* is a competency. Another very important metadata that exercises are required to be annotated with, is one of the five difficulty values: `very easy`, `easy`, `medium`, `difficult` or `very difficult`.

Out of a set of evidences, consisting of a concept, a cognitive process, a difficulty value and the achievement (whether the exercise was solved correctly), the student model estimates students' proficiencies. These evidences are generated after each approach to a problem. For each pair of a concept and a cognitive process (i.e. for each *competency*) defined in the metadata of an exercise, evidences are generated, propagated to related concepts (thus becoming *indirect evidences*) and stored in its specific containers. The modeled belief about the student's competency is then updated based on the available evidences, where indirect evidences have less influence on the resulting mastery value. Figure 2 illustrates this process.

1.2 Item Response Theory in SLM

The calculation of the probabilities for different masteries is based on the Item Response Theory (IRT) [Lord, 1980], which describes the relationship between the probability for a correct answer X to a mastery value (ability) θ:

$$P_i(X = correct|\theta) = c_i + \frac{1 - c_i}{1 + e^{-a_i \cdot (\theta - b_i)}} \qquad (1)$$

The resulting function is called the *item characteristic curve* (ICC). Equation 1 shows the three-parameter model of the item response theory. The parameters are exercise (item) specific where

- a_i is the item *discrimination factor*,
- b_i is the item *difficulty*,
- $c_i \in [0, 1]$ is the item *guessing probability*.

The item difficulty b_i defines the location of the mastery value θ where the probability of correct response is exactly 0.5, i.e. $P_i(X = correct|\theta = b_i) = 0.5$. Hence, the larger b_i is, the larger the mastery θ has to be to have a probability of at least 50%.

The discrimination factor defines the steepness of the S-shaped curve. Therefore, it describes how well it is differentiated between learners having masteries below the item difficulty ($\theta < b_i$) and those having masteries above the item difficulty ($\theta > b_i$). Figure 3 shows different ICCs with differing item difficulties.

The actual domain of the mastery θ, the item discrimination factor a_i, and the item difficulty (or item location) b_i is $\mathbb{R}$. However, if one restricts the domain of the item difficulty b_i, then the most differing probabilities for different mastery values θ will be around that range as well. This is because of the nice property that $P_i(X = correct|\theta = b_i) = 50\%$. SLM makes use of this and restricts the domain of θ and b_i to $[0.0, 1.0]$, which simplifies the interpretation of difficulty values.

Furthermore, SLM restricts its calculation to those masteries θ which are "near" the item difficulty (i.e. $\theta \in [b_i - \delta, b_i + \delta]$, where δ is the *information radius*). This will keep the student model from assuming high mastery values for a student who solved easy exercises only.

For more details on how the student model of ACTIVEMATH works, please refer to [Faulhaber, 2007; Faulhaber

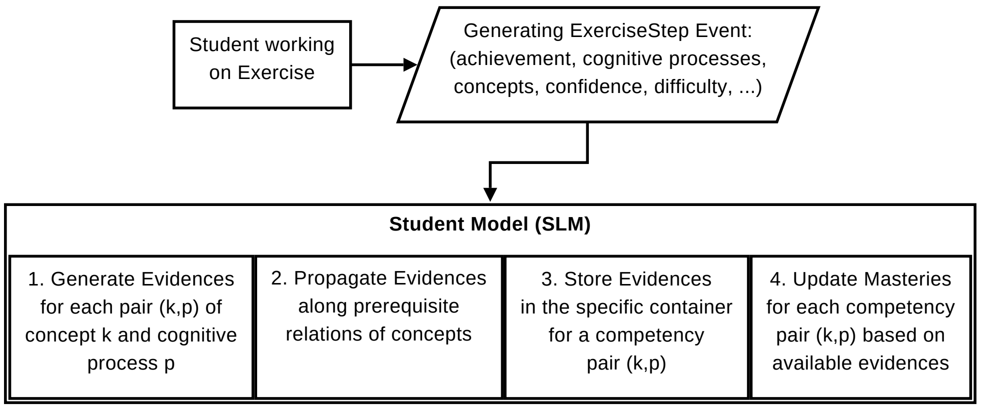

Student working on Exercise
Generating ExerciseStep Event: (achievement, cognitive processes, concepts, confidence, difficulty, ...)
Student Model (SLM)
1. Generate Evidences for each pair (k,p) of concept k and cognitive process p
2. Propagate Evidences along prerequisite relations of concepts
3. Store Evidences in the specific container for a competency pair (k,p)
4. Update Masteries for each competency pair (k,p) based on available evidences

Figure 2: The interaction of a student with ActiveMath creates events, which are handled by the student model.

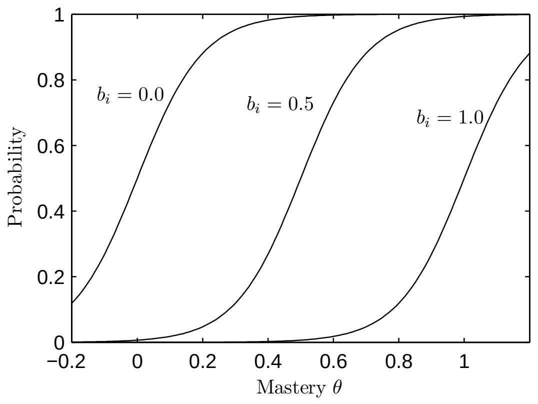


Figure 3: Typical ICCs showing probabilities of a correct answer for different masteries θ and varying item difficulties b_i. The guessing probability c_i is fixed to 0 and a_i to 10.

and Melis, 2008; Doost, 2010].

2 Which SLM Parameters can be Adjusted?

Depending on the structure, design and implementation of a student model, different ways of adjustments and optimization possibilities emerge. This section presents most of the student model parameters that have been selected for the optimization process. Some of the parameters were actually hard coded into the system. In order to discover them, a thorough analysis of the implementation was necessary.

Propagation Depth defines the maximum distance an evidence is propagated along the graph of prerequisites for concept relations. Too deeply propagated evidences might cause the SLM to transfer the student's knowledge about one concept to other possibly too little related concepts. However, with a completely deactivated propagation we might lose possible accuracy gains for concepts for which direct evidences are not available yet.

Weighing Evidences. How strong shall we weigh direct evidences compared to indirect ones when both are available? Giving no weight to indirect evidences is equal to having no propagation. However, giving them too much weight might understate the significance of direct evidences.

Evidence Container Size is the maximum number of stored direct and indirect evidences per competency and represents the *forgetting factor*. When this number is reached, old evidences are discarded in favor of new ones.

Difficulty Mapping. What is the range $[\alpha, \beta]$ into which the five difficulty values are mapped (every exercise in ACTIVEMATH is annotated with one out of five possible values). Probabilities received from the IRT strongly depend on the difficulty of an exercise. An arbitrary mapping might lead to misinterpretations of difficulty metadata and negatively effect the mastery estimations.

Default Discrimination Factor determines the curvature of the sigmoid function, which converges to a step function for high values making marginal masteries more probable and mid-level masteries less probable.

3 Random-Restart Hill Climbing

Hill climbing [Russel and Norvig, 2003] is a greedy local search algorithm and can be used for optimization problems. Hill climbing algorithms can find reasonable solutions in large or infinite (continuous) state spaces for which systematic algorithms fail.

Generally, when looking for a maximum of a function (optimization problem), the hill climbing algorithm works as follows:

1. Start at an arbitrary point
2. Calculate values for neighboring points
3. Move to the point with increased value
4. Terminate if no higher value could be found, otherwise continue at 1

The standard problem with this algorithm is that it may not find the optimal solution (i.e. the global maximum), but only a local maximum. However, with an extension known as random-restart one can increase the probability to find a global maximum considerably [Russel and Norvig, 2003]. Starting the hill climbing algorithm over and over again each time with randomly chosen initial states and saving only the maximum of the new values improves the probability of finding the global maximum.

The complex structure of the student model and the possibly interrelated parameters make it difficult - if not practically impossible - to determine the optimal values for the

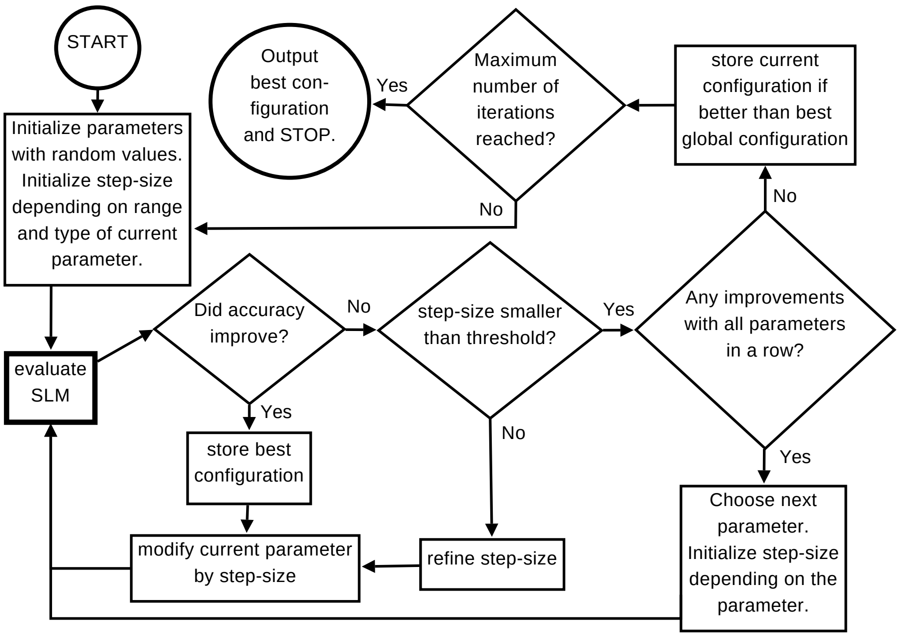

START
Initialize parameters with random values. Initialize step-size depending on range and type of current parameter.
evaluate SLM
Did accuracy improve?
No
Yes
store best configuration
modify current parameter by step-size
step-size smaller than threshold?
No
Yes
refine step-size
Any improvements with all parameters in a row?
No
Yes
Choose next parameter. Initialize step-size depending on the parameter.
store current configuration if better than best global configuration
Maximum number of iterations reached?
Yes
No
Output best configuration and STOP.

Figure 4: A flowchart of the modified RRHC algorithm used for parameter optimization.

parameters by classic machine learning methods. For optimizing the parameters mentioned in Sect. 2, the Random-Restart Hill Climbing (RRHC) algorithm has been modified. Our version steadily switches between the several different parameters. It modifies each parameter as long as improvements could be found, then switches to the next and returns to that parameter again as soon as all other parameters were taken into account. We restart with a new initial random state if no parameter change yielded improvements and terminate after a predefined maximal number of iterations.

The continuous state space of parameters with real values requires additional attention. In order to find the maximum quickly, we need to make noticeable larger steps than the smallest possible difference. This is done by using an initial step size δ depending on the range size of possible values for a parameter. The larger the interval, the larger will be δ. The parameter is increased by δ as long as there is any improvement. Then, the step size is refined by a factor $0 < \alpha < 1$ until the step size gets smaller than a predefined threshold ε (we used 0.1 for α and 0.0001 for ε). We continue with negative step sizes starting from $-\delta \cdot \alpha$ and increase it by multiplying with α again until we reached a value larger than $-\varepsilon$.

In short, we first start with a rough search by modifying a parameter with larger steps, then we conclude with fine tuning by modifying a parameter with smaller and smaller steps until we reach a certain threshold for a minimum step size. A flowchart for our modified version of RRHC is presented in Figure 4.

4 Evaluation and Results

An important question remains: what is the best function for the optimization problem in order to find a good parameter configuration for a student model?

The student model can be evaluated by replaying it with empirical data collected during another experiment. Based on this data an experiment can be simulated for each possible configuration without organizing new ones. Its log data can be rerun by going through all event entries chronologically and passing them over to the student model. During the simulation process we can query the student model about its belief and compare it to the actual achievements before passing on this information to the student model. Figure 5 depicts a diagram of this evaluation process.

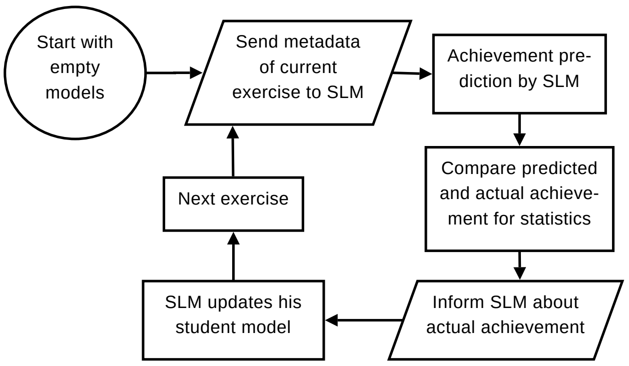

Start with empty models
Send metadata of current exercise to SLM
Achievement prediction by SLM
Compare predicted and actual achievement for statistics
Inform SLM about actual achievement
SLM updates his student model
Next exercise

Figure 5: The evaluation process of the student model SLM through replay of log data.

We believe that the more accurate the student model is, the better are the *achievement* predictions, i.e. whether a student is able to solve a specific exercise or not. Our function for the optimization problem is therefore the achievement prediction accuracy which we strive to maximize. It is calculated as shown in Figure 6, where n is the overall number of exercise steps available (i.e. the total number of predictions), ach_i is the actual achievement in the ith exercise step[2], and $pred_i$ is the student model's prediction for the achievement of the ith exercise step.

For the evaluation, log data collected in the context of the ATuF project [Eichelmann *et al.*, 2008] (with 190 students, 6th and 7th graders) and data from experiments of the ALoE project [3] [Tsovaltzi *et al.*, 2009] (with 77 students, mostly 6th graders) has been used.

[2]It is 1 if the student's answer was correct, otherwise 0.

[3]Funded by the German National Science Foundation (DFG) (ME1136/7).

$$\begin{aligned} accuracy &= \frac{\text{number of correct predictions}}{\text{overall number of predictions}} \\ &= \frac{1}{n} \cdot \sum_{i=1}^{n} 1 - |ach_i - pred_i| \end{aligned}$$

Figure 6: Achievement Prediction Accuracy

4.1 Results

For the ATuF data, the original SLM had an achievement prediction accuracy of 79%. This is already a highly improved and optimized value, which was the result of several years of experimentation. Still, with the parameter configuration obtained from the presented method, the accuracy became 82%. In absolute numbers, this means that 470 out of 15680 predictions which were wrongly predicted before are now predicted correctly. For content selection based on the estimated competencies, every wrongly selected item (like selecting too easy or too difficult exercises, or leaving missing prerequisites away) could make considerable differences.

A statistical hypothesis test (one-sided) with a statistical significance of 5% and the null hypothesis being the non-improvement of the prediction accuracy yields a z-score of -6.3. This means that we have to reject the null hypothesis (since the z-score is less than -1.65) and assume that the alternative hypothesis is true, namely that the accuracy improvement by 3% is *(strongly)* significant. The evaluation for the second data set, the ALoE data, revealed for the original SLM an overall prediction accuracy of 69%[4]. With the optimized parameter configuration the overall prediction accuracy rose by 2% to 71%. This yields a z-score of 2.1, which is statistically significant as well.

In the following, the single parameter values of the configuration which yielded the highest accuracy will be mentioned briefly. Surprisingly, a *propagation depth* of one produced almost the same accuracy as no propagation. However, a depth of two or more worsened the result. This would mean that we cannot really transfer the students' knowledge level of one concept to related concepts except to directly adjacent ones (without any gain, though). For *weighing* direct and indirect evidences a ratio of 12: 1 was obtained. As expected, the value of direct evidences is considerably higher. Furthermore, keeping a maximum of six *evidences* per competency produced the highest accuracy. So did the interval $[0.14, 0.86]$ for the *difficulty mapping*, which means that the easiest exercises get a difficulty value of 0.14 instead of 0. Finally, a *discrimination factor* of 9.6 showed to be best.

5 Related Work

Finding optimal parameter configurations is a problem many student models have to tackle. The parameters may belong to Bayesian networks, to metadata of exercises or concepts (also known as rules), to the working mechanism of a student model itself or to any other part related to the student model. Corbett and Anderson conducted several experiments to evaluate their knowledge tracing student model. From the resulting data they fit the weights for learning and performance parameters of each rule and student [Corbett and Anderson, 1995]. Cen et al. have shown that using an educational data mining technique called *Learning Factors Analysis* (LFA) they could improve the learning efficiency by 12% [Cen *et al.*, 2007]. For this, they optimized knowledge tracing parameters of the Cognitive Geometry Tutor based on older log data. In the experiment with the optimized model students required on average 12% less time to reach the same level.

Gong, Beck and Heffernan compared two techniques for student modeling, *Knowledge Tracing* and *Performance Factor Analysis* (PFA), in terms of their predictive accuracy and parameter plausibility [Gong *et al.*, 2010]. For fitting the knowledge tracing model, they used and compared two different techniques: *expectation maximization* and *brute force*. To minimize the problem of local maxima, they restart their fitting algorithm multiple times as well. In their work, Gong et al reported that the *brute force* method was not as good as the *expectation maximization* method. *Brute force* is a very expensive and – because of the continuous state space – an infeasible model fitting method. Random-restart hill climbing (RRHC) is a method in between *expectation maximization* and *brute force*. In a brute force manner it evaluates and tries out different parameter configurations. However, it does not evaluate an equally distributed sampled set of configurations, but instead it modifies parameters stepwise in the direction of expected improvement until a maximum is reached.

Local search methods, like hill climbing, are often used for finding solutions efficiently, especially for NP-hard problems like the traveling salesman problem [Aarts and Lenstra, 2003]. Random-restart or multiple-restart is a known solution for increasing the probability to obtain a global maximum instead of a local one. The usage of dynamically varying step sizes is also not new: it has been already reported in [Yuret and de la Maza, 1993; Miller, 2000]. Basically, any local search algorithm could have been used. RRHC, however, is because of its simplicity and straight forwardness the first choice for being adopted for optimizing several student model parameters. Other possible techniques are, for example, simulated annealing or genetic algorithms.

This work used the achievement prediction accuracy to measure the performance of student models. This is a common approach. Corbett and Anderson, for example, used the prediction accuracy to evaluate their knowledge tracing student model used in the ACT Programming Tutor [Corbett and Anderson, 1995]. Their performance prediction after every step depends on (1) the probability that a rule is in the learned state, (2) a slip parameter and (3) a guess parameter for the according rule. Desmarais and Gagnon compared the prediction accuracy of two cognitive student models: one based on a standard Bayesian network approach and the other on a more constrained one [Desmarais and Gagnon, 2006]. Previous versions of student models of ACTIVEMATH were evaluated by the prediction accuracy as well [Faulhaber, 2007; Faulhaber and Melis, 2008].

6 Conclusions and Future Work

This paper presented how the parameters of ACTIVEMATH's student model had been optimized with the local search algorithm known as random-restart hill climbing. Several parameters were presented, which were then optimized by a modified version of the RRHC algorithm.

[4]The reason why the prediction accuracy on the ALoE data is on average about 10% lower compared to the ATuF data is that the metadata quality of the ALoE data is worse and the number of difficult exercises is higher.

The modified RRHC uses dynamic step sizes and steadily switches between different parameters. The evaluation function replays log data to measure the achievement prediction accuracy of a configuration. The achievement prediction accuracy improved significantly by 3% for the ATuF data and by 2% for the ALoE data.

Until this work, the parameters of ACTIVEMATH's student model remained to be a manually chosen configuration. The evaluated parameter configuration did indeed improve SLM's accuracy. Since the performance of the original SLM was already quite good, an increase by 2% and 3% is an acceptable improvement. Nevertheless, improvements by 5% and above would have been more convincing.

The presented methodology can be generalized to optimize other user models as well. First of all, the set of parameters which have to be optimized has to be fixed. Additionally, an evaluation method has to be defined — replaying log data and measuring the prediction accuracy is just one possibility. Finally, RRHC can be used to find an optimized parameter configuration.

In the future we plan to use log data obtained from other projects using different exercises with participants of the same age as well as more experienced participants. This might help us to find out whether the obtained configuration is specific to the young learners or to the used exercises domain (which was fraction mathematics in both experiments, ATuF and ALoE). We want to further analyze the necessity of propagation and possible differences between age groups.

Acknowledgements

This article results from the ATuF project (ME 1136/5-2) funded by the German National Science Foundation (DFG).

References

[Aarts and Lenstra, 2003] Emile L. Aarts and Jan K. Lenstra. *Local search in combinatorial optimization.* Princeton Univ Pr, 2003.

[Cen *et al.*, 2007] Hao Cen, Kenneth R Koedinger, and Brian Junker. Is over practice necessary? –improving learning efficiency with the cognitive tutor through educational data mining. In *Proceeding of the 2007 conference on Artificial Intelligence in Education*, pages 511–518, Amsterdam, The Netherlands, 2007. IOS Press.

[Corbett and Anderson, 1995] Albert T Corbett and John R Anderson. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User Modeling and User-Adapted Interaction*, 4:253–278, 1995.

[Desmarais and Gagnon, 2006] Michel C Desmarais and Michel Gagnon. *Bayesian Student Models Based on Item to Item Knowledge Structures*, pages 111–124. Springer-Verlag Berlin Heidelberg, 2006.

[Doost, 2010] Ahmad Salim Doost. Enhancement of activemath's student model. Master's thesis, Saarland University, 2010.

[Eichelmann *et al.*, 2008] Anja Eichelmann, Susanne Narciss, Arndt Faulhaber, and Erica Melis. Analyzing computer-based fraction tasks on the basis of a two-dimensional view of mathematics competences. In *EARLI SIG 6&7*, pages 125–134. Springer, 2008.

[Faulhaber and Melis, 2008] Arndt Faulhaber and Erica Melis. An efficient student model based on student performance and metadata. In Nikos Avouris, Nikos Fakotatis, Constantine D. Spyropoulos, and Malik Ghallab, editors, *Proceedings of 18th European Conference on Artificial Intelligence. 18th European Conference on Artificial Intelligence (ECAI-08), July 21-25, Patras, Greece*, volume 178 of *Frontiers in Artificial Intelligence and Applications, FAIA*, pages 276–280. University of Patras, IOS Press, 2008.

[Faulhaber, 2007] Arndt Faulhaber. Building a new learner model for activemath combining transferable belief model and item response theory. Master's thesis, Saarland University, 2007.

[Flynt and Morton, 2009] Samuel W Flynt and Rhonda Collins Morton. The teacher shortage in america: Pressing concerns. *National Forum of Teacher Education Journal*, 19(3), 2009.

[Gong *et al.*, 2010] Yue Gong, Joseph E Beck, and Neil T Heffernan. *Comparing Knowledge Tracing and Performance Factor Analysis by Using Multiple Model Fitting*. 2010.

[Ingersoll and Perda, 2009] Richard M Ingersoll and David Perda. The mathematics and science teacher shortage: Fact and myth. Technical Report CPRE Research Report #RR-62, The Consortium for Policy Research in Education, March 2009.

[Kohlhase, 2006] Michael Kohlhase. *OMDoc – An Open Markup Format for Mathematical Documents [version 1.2]: Foreword by Alan Bundy*. Lecture Notes in Computer Science. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[Lord, 1980] Frederic M. Lord. Applications of item response theory to practical testing problems. Hillsdale NJ: Erlbaum, 1980.

[Miller, 2000] Ronald E Miller. *Optimization: Foundations and Applications*. John Wiley & Sons, 2000.

[Russel and Norvig, 2003] Stuart Russel and Peter Norvig. *Artificial Intelligence: A Modern Approach*, pages 110–114. Prentice Hall, 2 edition, 2003.

[Tsovaltzi *et al.*, 2009] Dimitra Tsovaltzi, Erica Melis, Bruce McLaren, Michael Dietrich, George Goguadze, and Ann-Kristin Meyer. Erroneous examples: A preliminary investigation into learning benefits. In Ulrike Cress Vania Dimitrova Marcus Specht, editor, *Proc. of the First European Conference on Technology Enhanced Learning (EC-TEL 2009)*, volume LNCS 5794 of *Lecture Notes in Computer Science*, Heidelberg, 2009. Springer-Verlag.

[Ullrich and Libbrecht, 2008] Carsten Ullrich and Paul Libbrecht. *Educational Services in the ActiveMath Learning Environment*, pages 211–236. The Future of Learning. IOS Press Amsterdam, March 2008.

[Yuret and de la Maza, 1993] Deniz Yuret and Michael de la Maza. Dynamic hill climbing: Overcoming the limitations of optimization techniques. In *In The Second Turkish Symposium on Artificial Intelligence and Neural Networks*, pages 208–212, 1993.