Modeling and Predicting Students Problem Solving Times*

P. Jarušek and R. Pelánek

Faculty of Informatics, Masaryk University Brno

Abstract. Artificial intelligence and data mining techniques offer a chance to make education tailored to every student. One of possible contributions of automated techniques is a selection of suitable problems for individual students based on previously collected data. To achieve this goal, we propose a model of problem solving times, which predicts how much time will a particular student need to solve a given problem. Our model is an analogy of the models used in the item response theory, but instead of probability of a correct answer, we model problem solving time. We also introduce a web-based problem solving tutor, which uses the model to make adaptive predictions and recommends problems of suitable difficulty. The system already collected extensive data on human problem solving. Using this dataset we evaluate the model and discuss an insight gained by an analysis of model parameters.

1 Introduction

Problem solving is an important component of education. To make problem solving activities attractive, it is important to confront students with problems of suitable difficulty – neither too easy, nor too difficult (see the flow concept [3, 4]). Since students vary in their skills, it is crucial to make problem recommendations individually adaptive.

Our main aim in this paper is to predict a difficulty of problems, more specifically to predict a time it will take a student to solve a problem. We aim to do the prediction based on previous data about problem solving activity of this and other students. To this end we model a relation between a problem solving ability and a time to solve a problem. As a concrete application of the proposed model we develop a problem solving tutor – an online application for enhanced learning.

To make our setting clear, we describe one of the problems that we use in our tutoring application (and also in evaluation in this paper). The goal of the problem "Graphs and functions" (see Fig. 1) is to identify a formula for describing a function, which is specified by its graph. As a tool for solving students may use interactive graph drawing facility which plots their attempts to the graph[1]. By solving these puzzles students train their ability to visualize math functions and deduct formulas from visualizations.

* This work is supported by GAČR grant no. P202/10/0334.

[1] The reader can try the problem at `tutor.fi.muni.cz`.

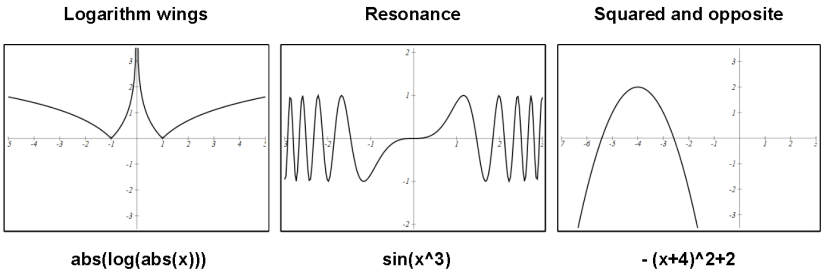

Fig. 1. Three instances of the problem "Graphs and functions" and their solutions. Note that a title is sometimes used to give students a hint.

Our work is related to four research areas, but has significant differences from each of them. *Item response theory* (IRT) [2, 5] is used particularly for computerized adaptive testing (i.e., for measuring student latent ability). IRT considers tests where each question (item) has several possible answers. IRT models give relation between student ability and a probability of a correct answer. Our model is directly inspired by IRT, but there is an important difference. The IRT focuses on tests with correct and incorrect answers, whereas we study problem solving and measure a time to solve a problem (as illustrated above on the "Graphs and functions" problem). The most relevant aspect of IRT are models of response times [15, 16] (which are discussed in more detail in Section 2.3). Unlike IRT which is primarily applied for adaptive testing, we are interested in intelligent tutoring.

Intelligent tutoring systems [1] are computer programs used to make learning process more adaptive and student oriented. They provide background information, problems to solve, hints, and learning progress feedback. Well known example of an intelligent tutoring system is a system for teaching algebra [9, 8]. Tutoring systems usually have static structure which is determined by an expert in a particular domain. Our system is dynamic and recommends problems based on collected problem solving data. Most research on tutoring systems focuses on the "inner loop" (how to give hints about a problem), we focus solely on the "outer loop" (how to dynamically order problems) [17].

Recommendation systems [7] are used mainly in e-commerce. These systems recommend to users products that may be interesting for them (e.g., books on Amazon, films on Netflix). One of the approaches to recommendation – collaborative filtering – is based on the use of data on user behaviour. With this approach a recommender system at the same time collects data and uses these data to make predictions and recommendation. We build our system in the same way, although we are not interested in recommending products, but problems of suitable difficulty. This approach is in contrast with the mainly linear approach (collect data, calibrate models, use models) used in IRT and in education in general.

Research in *human problem solving* [14] so far focused mainly on analysis and computational modeling of human problem solving for a particular problem, e.g., Tower of Hanoi [10], Chinese ring puzzle [11], Fifteen puzzle [13], Sokoban [6], Sudoku [12]. This research provides insight into problem difficulty, but the insight is usually closely tied to a particular problem.

We combine principles from the above mentioned areas in the following way. We build *tutor* for practising *problem solving* which *recommends* users problems of suitable difficulty, based on the previously collected data and using a novel model, which is a variation on models used in the *item response theory*.

Our specific contributions are the following. We propose a general setting for modeling problem solving times and three specific models. For these models we discuss methods for parameter estimation and provide evaluation on large problem solving data. We also present direct application of the model – a problem solving tutor (`tutor.fi.muni.cz`).

2 Modeling Problem Solving Times

In this section we describe our approach to modeling problem solving times. It is analogical to models used in the item response theory, but instead of modeling probability of a correct answer, we model a time to solve a problem.

2.1 Summary of Item Response Theory

We start by summarising main principles of the item response theory. We focus only on aspects relevant to our model and we provide simplified description of the theory. The item response theory is a tool for designing, analyzing, and scoring tests, questionnaires, and similar instruments that measure abilities [2]. One of its main applications is computerized adaptive testing – selection of questions in test is done dynamically based on answers of a student.

Main assumption of IRT is that a given test measures one latent ability θ. IRT models response to one item (test question) as a relation between this ability θ and probability P that the item is correctly answered (basic models consider only dichotomous questions). This relation is expressed by an item response function. The basic model in IRT is 3 parameter logistic model (see also Fig. 2):

$$P_{a,b,c,\theta} = c + (1 - c)\frac{e^{a(\theta-b)}}{1 + e^{a(\theta-b)}}$$

This model has three parameters: b is the basic difficulty of the item, a is the discrimination factor (slope of the curve, how well the item discriminates based on ability), and c is the pseudo-guessing parameter (lower limit of the curve, probability that even a student with very low ability will guess the correct answer). Other two often used models are derived from this model by fixing values of some parameters: a two parameter logistic model ($c = 0$) and a one parameter logistic model ($c = 0, a = 1$).

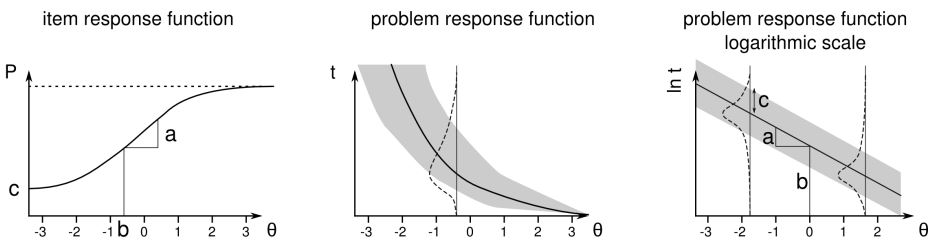

Fig. 2. Intuitive illustration of item response function, general problem response function, and a specific problem response function under our assumptions. Dashed lines illustrate distributions at certain points; solid lines denotes the median of a time distribution, grey areas depict the area into which most attempts should fall.

To apply these models, it is necessary to estimate values of their parameters. Since we do not know neither the item parameters (a, b, c), nor student's abilities (θ), we need to estimate both of these at the same time. This is usually done by joint maximum likelihood estimation, which proceeds by repeating two steps: estimating abilities from item parameters and estimating item parameters from abilities. These steps are repeated until parameter values converge.

An important feature of IRT models is group invariance – item parameters do not depend on a subset of students which answered the item, i.e., even if some item is answered only by above-average students, the estimated item parameters should be similar as if the item was answered by a representative subset of students.

2.2 Problem Solving Times

There are many extensions of the basic IRT models, e.g., models for items with polytomous answers or models which take into account response times. But none of these models is directly applicable to the problem solving setting. We propose a model, which relates problem solving ability and a time to solve a problem. At the moment we study only students time to solve a problem and not a quality of solutions (i.e., the current theory is applicable only to well-structured problems with easily verifiable solutions like the "Graphs and Functions" problem in Fig. 1).

The basic principles are analogical to the above mentioned principles of IRT. Similarly to IRT, we assume that a problem solving performance depends on one latent problem solving ability θ. We are interested in "problem response function" $f(\theta)$, which for a given ability θ gives an estimate of a time to solve a problem. More specifically, the function gives a probabilistic density of times. Fig. 2 gives a comparison of basic setting of IRT and our model.

2.3 Specific Assumptions and Model

To obtain a specific model we make the following two assumptions. First, the distribution of solving times $f(\theta)$ for students with a fixed ability θ is a log-

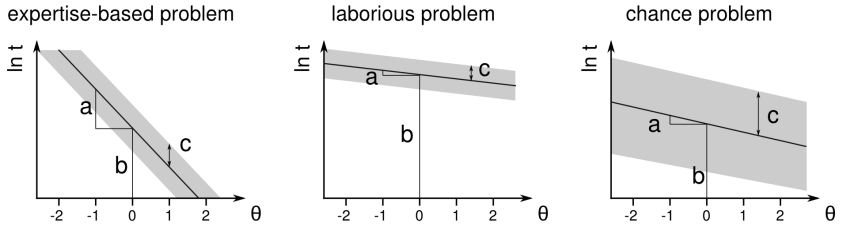

Fig. 3. Examples of different problem types and their modeling using the 3 parameter model. See Fig. 6 for specific examples.

normal distribution. Second, the mean and variance of the distribution $f(\theta)$ are exponentially dependent on θ (this dependency can be, of course, changed by rescaling θ; we implicitly assume that problem solving ability is normally distributed in the population).

These assumptions are grounded on our data about human problem solving from our previous experiments [6, 12] and on experience in modeling response times in IRT [16]. Moreover, the assumptions lead to a tractable model with nice properties – by using logarithm of time we obtain linear relationship and normal distributions.

Based on these assumptions, we can now specify a concrete model. Our basic model is a 3 parameter model in which the intuitive meaning of the parameters is the following (we intentionally use notation analogical to IRT): discrimination factor a, basic difficulty of the problem b, randomness factor c.

The problem response function, i.e., the probability that a student with ability θ will solve a problem at a logarithm of time t, is given by a normal distribution with a mean $b + a\theta$ and a variance c^2. Thus we have:

$$f_{a,b,c,\theta}(\ln t) = \mathcal{N}(a\theta + b, c)(\ln t) = \frac{1}{\sqrt{2\pi}c} e^{-\frac{(\ln t - (a\theta + b))^2}{2c^2}}$$

This model and intuition behind its parameters are illustrated in Fig. 2. Discrimination factor a describes the slope of the function, i.e., it specifies how the problem distinguishes between students with different ability. Basic difficulty describes expected solving time for student with average ability. Randomness factor describes variance in solving times for particular ability. The model is relatively simple, yet it can capture different aspect of problem difficulty and their combinations (see Fig. 3).

The presented model is not yet identified as it suffers from the "indeterminacy of the scale" issue in the same way as the basic IRT model, e.g., we can multiply θ and divide a by k without any effect on the model. Thus we further require the following normalization – for a set of problem parameters b_i, a_i, c_i and student parameters θ_j, we require that the mean of θ_j is 0, the mean of a_i is -1.

Similarly to IRT, we can also use simplified models. A two parameter model is obtained by using a constant randomness factor k, a one parameter model is obtained by using constant randomness and discrimination factors:

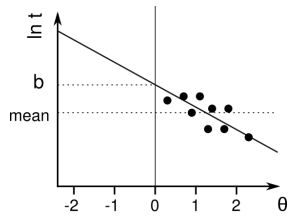

Fig. 4. If a problem is solved by above-average persons, the mean time underestimates the difficulty of a problem, whereas our model can capture it correctly.

$$f_{a_i,b_i,\theta}(\ln t) = \mathcal{N}(b_i + a_i\theta, k)(\ln t) \qquad f_{b_i,\theta}(\ln t) = \mathcal{N}(b_i - \theta, k)(\ln t)$$

Our model is similar to van der Linden's model for response times in IRT [15]. There are two main differences. First, he uses the model in a context of testing, where the timing information is just supplementary to information about correctness of an answer, whereas in our case timing is the main focus. Second, his model has just two parameters (basic difficulty and randomness).

2.4 Group Invariance

An important feature of the approach is that the models are group invariant (analogically to IRT), i.e., parameters of a problem do not depend on a subgroup of students which solve the problem.

Let us explain this important feature by comparing our 1 parameter model to the baseline metric of a problem difficulty – the mean time to solve the problem (Fig. 4). In both cases the problem difficulty is captured by one number – by difficulty parameter b in our model or by the mean m. If we have a set of problems, then it typically happens that harder problems are solved only by students with above-average ability. In this case the mean time underestimates the real difficulty of the problem, whereas our mode is not biased by the selection of students.

3 Parameter Estimation

Since we do not know neither parameters of problems, nor parameters of students, we need to estimate them. To compute these estimates we use data of the following type: problem i was solved by a student j in time t_{ij}. From these data we need to estimate both problem parameters a_i, b_i, c_i and student parameters θ_j.

One way to do this is to apply a generic data-fitting method like non-linear least squares directly on the model and to use existing software implementations to compute estimates. Here we discuss an alternative iterative approach

which is analogical to the joint maximum likelihood calculation in IRT. Advantage of the iterative approach is that it computes estimates for each student (problem) separately from others, so it is possible to update estimates locally without recomputing the whole set of parameters – this is a useful feature for the application of the approach in our problem solving tutor, which needs to make prediction in realtime. Moreover, the iterative approach gives better insight into the computation.

3.1 Estimating Ability

Suppose that a student solved n problems, where i-th problem has parameters a_i, b_i, c_i and was solved in time t_i. Based on these data we want to estimate the ability θ of the student. We do this by finding a maximal likelihood θ. The likelihood of the observed times $t_1, \ldots, t_n$ given our 3 parameter model is:

$$L = \prod_{i=1}^{n} f_{a_i,b_i,c_i,\theta}(\ln t_i) = k \prod_{i=1}^{n} \frac{1}{c_i} e^{-\frac{(\ln t_i - (b_i + a_i\theta))^2}{2c_i^2}}$$

We need to find the value of θ such that L is maximized. As is usual, we proceed by finding maximum of $\ln L$ (which is the same as maximum of L):

$$\ln L = k + \sum_{i=1}^{n} \ln \frac{1}{c_i} + \frac{1}{2c_i^2}(a_i^2\theta^2 + 2a_i\theta(\ln t_i - b_i) + (\ln t_i - b_i)^2)$$

Since this is a quadratic function in θ, we can find maximum by finding the value of θ for which the derivation is zero:

$$\frac{\ln L}{\partial\theta} = \sum_{i=1}^{n} \theta\frac{a_i^2}{c_i^2} + \frac{a_i}{c_i^2}(\ln t_i - b_i) = 0 \qquad \theta = \frac{\sum_{i=1}^{n} \frac{a_i^2}{c_i^2}\frac{\ln t_i - b_i}{a_i}}{\sum_{i=1}^{n} \frac{a_i^2}{c_i^2}}$$

The resulting expression for θ has a clear intuitive interpretation. The expression $(\ln t_i - b_i)/a_i$ is a local estimate of ability for i-th problem – it is the value of θ for which the expected logarithm of time is $\ln t_i$. The overall estimate of θ is obtained as a weighted average of these local estimates, where the weight is given by the expression a_i^2/c_i^2, i.e., the more discriminating and less random a problem is, the more weight it gets (which is exactly what one would intuitively expect). For the one parameter model model this expression simplifies to $\theta = (\sum_{i=1}^{n} b_i - \ln t_i)/n$.

3.2 Estimating Problem Parameters

Suppose that a problem was solved by n students, where j-th student has ability θ_j and solved the problem in time t_j. Now we want to estimate problem parameters a, b, c. Maximal likelihood estimates can be found by a regression analysis. For the two and three parameter models we can use standard linear regression (least square method), because for our model (linear dependence with normally

distributed errors) the least square method gives maximal likelihood estimation. Parameter c is then estimated from error residuals.

For the one parameter model we are looking for linear regression line with a fixed slope $a = -1$, thus we need to minimize the following sum of squares: $\sum_{j=1}^{n}(\ln t_j - (b - \theta_j))^2$. This is a quadratic function with a minimum at $b = (\sum_{j=1}^{n} \ln t_j + \theta_j)/n$.

3.3 Joint Estimation

So far we assumed that either abilities are known exactly and we estimate problem parameters, or that problem parameters are known exactly and we estimate student ability. In reality, of course, we do not known exactly neither student abilities nor problem parameters. We compute their estimates by an iterative bootstraping process:

1. initialization: for each problem i, set problem parameters as follows: $a_i = -1$, $b_i =$ mean time, $c_i = k$,
2. repeat until a selected convergence criterion is satisfied:
   (a) for each user j update the estimates of θ_j based on the current problem parameters,
   (b) for each problem i update the estimates of a_i, b_i, c_i based on the current ability estimates.

Although each of the steps computes maximum likelihood estimates (with respect to fixed input parameters), overall it is only approximation of the joint maximum likelihood. One of the reasons is that the input parameters in each step of iteration are only estimates and they differ in their confidence, e.g., for students which solved more problems we have better estimates of their ability. However, this aspects is not included in the described computation. This issue can be (pragmatically) addressed by using weighted least squares for estimating parameters a_i and b_i with weight for each student dependent on the number of solved problems.

4 Application and Evaluation

We apply the model in development of a web portal for tutored problem solving: a "Problem solving tutor". In this section we describe the system and provide evaluation of the model using the collected data.

4.1 Problem Solving Tutor

Problem solving tutor is a free web-based tutoring systems for practicing problem solving skills; the system is available at `tutor.fi.muni.cz`. The tutor contains large set of problems of different types. At the moment the system contains a math problem, two programming exercises, regular expressions, and 10 logic puzzles. For each problem type there are between 30 and 80 instances of different

difficulty. The system is in active development and we are continuously adding new problems and collecting more data. Although the system was made public only recently, it is already used by several high schools and has more than 1 200 registered users who have spent more then 2000 hours solving more than 60 000 problems.

At the moment we focus solely on the "outer loop" of the tutor [17], i.e., recommending problem instances of the right difficulty. The inner loop (within a selected problem) is currently not present – we just let the users solve a problem, either they finish or give up; i.e., rather than giving hints we give users different (simpler) problem to solve.

The following modules provide the core functionality of the system:

- *Problem simulators.* Simulators provide web interface for solving individual problems (puzzles).
- *Prediction module.* Based on the collected data it makes predictions about solving time for given user.
- *Recommendation module.* Based on predictions it recommends to a user an unsolved problem of suitable difficulty. Recommendations are based on the predicted times and on the session history (e.g., number of recent successes and failures).
- *Feedback module.* Based on the collected data it gives a user comparison with other users; particularly we provide immediate feedback after finishing problem solving (to support the flow phenomenon [4]).

Our focus at the moment is on the prediction module, which implements the model described above. The prediction module uses the iterative process for computing the parameter estimates. It would be computationally expensive to recompute all parameter estimates after each solved problem. Thus when user j solves problem instance i we update only parameters a_i, b_i, c_i, θ_j and only occasionally we run complete update of all parameters (i.e., full run of the iterative computation).

For every solved problem instance we store not only a final solving time, but also every performed move. In this way we collect extensive data about human problem solving. These data may be useful for further analysis of human problem solving behaviour and more detailed research into problem difficulty (in a similar way as in our previous research [6, 12]).

4.2 Evaluation of Predictions

We have evaluated two approaches for estimating parameters: our implementation of the iterative estimation process (as described above) and estimation using a generic non-linear least squares method (using R software). The iterative computation converges very quickly (for practical use 3 iterations are enough) and both methods provide very similar results.

Student abilities θ are approximately normally distributed and the relation between ability and logarithm of time is nearly linear, see Fig. 6. These results support assumptions on which the model is based (see discussion in Section 2.3).

Table 1. Evaluation of quality of predictions measured by Spearman correlation coefficients.

Problem	Baseline	Model	Improvement (%)
Robot Karel	0.45	0.80	77.78
Nurikabe	0.42	0.73	73.81
Regular expressions	0.47	0.73	55.32
Graphs and functions	0.51	0.77	50.98
Slitherlink	0.64	0.84	31.25
Sokoban	0.67	0.80	19.40
Tents puzzle	0.58	0.67	15.52
Robot programming	0.62	0.71	14.52
Tilt maze	0.71	0.78	9.86
Region division	0.52	0.57	9.62
Rush Hour puzzle	0.78	0.84	7.69
Number maze	0.78	0.82	5.13

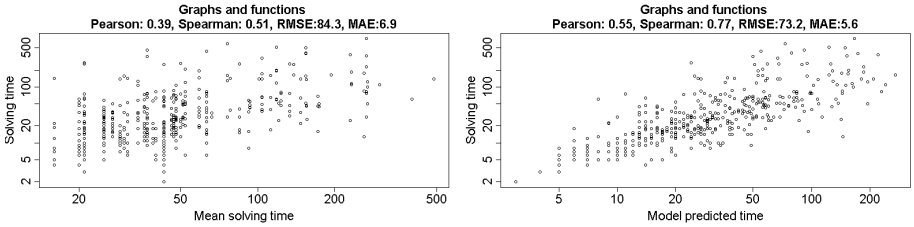

Fig. 5. Prediction versus real problem solving data: comparison of prediction based on mean time (left) and on the one parameter model (right).

To evaluate a quality of predictions, we compare predictions based on the one parameter model with the baseline metric "mean time to solve a problem". Both prediction methods were trained using 90% of data and evaluated on the remaining 10% of data. Fig. 5 shows predictions and solving times for the Graphs and functions problem. Table 1 compares the results using the Spearman correlation coefficient. We have also evaluated other metrics like the Pearson correlation coefficient, root mean square error and mean absolute error, the results are similar.

As the table shows, the model leads to improvement of 5-80% in precision of the prediction. As expected, the improvement is modest in cases of simple puzzles which contain a "luck factor" (e.g., mazes), and it is more pronounced for problems, where problem solving skill plays significant role (pedagogical problems or more complex logic puzzles like Nurikabe or Slitherlink).

There is no significant difference between quality of predictions based on the one parameter model and the three parameter model. We suppose that the three parameter models needs more data to make a difference for predictions. Nevertheless, even with current data the three parameter model brings an additional insight – as we now illustrate on one of the examples.

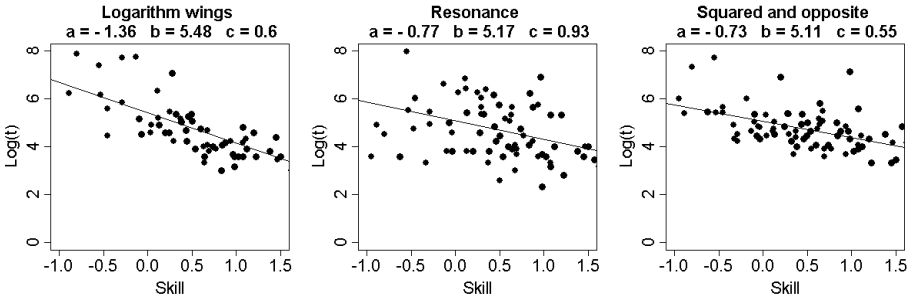

Fig. 6. "Graphs and functions" problem – three specific examples, for each of them we provide the collected data and values of parameters of the three parameter model.

4.3 Insight Gained from Parameter Values

Let us illustrate the insight gained from the values of three parameter model on the problem "Graphs and functions", which is described in introduction. Fig. 6 shows the collected data and values of model parameters for the three examples illustrated in Fig. 1. All three problems have similar basic difficulty (parameter b), but they differ in the other two parameters. The "Logarithm wings" problem has small randomness and large discrimination; the "Resonance" problem has large randomness and small discrimination; and the "Squared and opposite" problem has small randomness and small discrimination.

These parameters provide a valuable insight, which can be potentially used for further improving intelligent tutoring systems. Problems with small discrimination and large randomness clearly depend more on luck than on ability and thus are probably not a very good pedagogical examples (so we may want to filter out such examples). At the beginning of the problem solving session (when we do not have a good estimate of student ability), we may prefer problems with small discrimination (so that we have higher confidence in solving time estimation), later we may prefer problems with higher discrimination (so that we select problems "tuned" for a particular student).

5 Conclusions and Future Work

We propose a novel variation on the item response theory where we do not focus on correctness of answers but on the time to solve a problem. Our model is given by a function which for a problem solving ability gives a probabilistic distribution of time to solve a problem. We provide a specific model with three parameters and discuss methods for parameter estimation. We evaluate the model and apply it in a problem solving tutor, which is already used in education.

This work lays foundations for future work in several directions. First, it would be useful to extend the model to deal with unfinished attempts (when a student spends some time trying to solve a problem and then abandons the

problem, we do not include this information in our computation, although it can plausibly improve our parameter estimates). Second, the problem solving tutor can be extended by including an inner loop (hints for problem solving), and more sophisticated recommendations (e.g., using session history). Third, the collected data could be used to analyse causes of difficulty of particular problems (in a similar way as in our previous work [6, 12]).

References

1. J.R. Anderson, C.F. Boyle, and B.J. Reiser. Intelligent tutoring systems. *Science*, 228(4698):456–462, 1985.
2. F.B. Baker. *The basics of item response theory*. University of Wisconsin, 2001.
3. M. Csikszentmihalyi. *Beyond boredom and anxiety*. Jossey-Bass, 1975.
4. M. Csikszentmihalyi. *Flow: The psychology of optimal experience*. HarperPerennial New York, 1991.
5. R.J. De Ayala. *The theory and practice of item response theory*. The Guilford Press, 2008.
6. P. Jarušek and R. Pelánek. What determines difficulty of transport puzzles? In *Proc. of Florida Artificial Intelligence Research Society Conference (FLAIRS 2011)*, 2011.
7. P.B. Kantor, F. Ricci, L. Rokach, and B. Shapira. *Recommender systems handbook*. Springer, 2010.
8. K.R. Koedinger, J.R. Anderson, W.H. Hadley, and M.A. Mark. Intelligent tutoring goes to school in the big city. *International Journal of Artificial Intelligence in Education*, 8(1):30–43, 1997.
9. K.R. Koedinger, A.T. Corbett, S. Ritter, and L. Shapiro. Carnegie Learning's Cognitive Tutor: Summary research results. *White paper. Available from Carnegie Learning Inc*, 1200, 2000.
10. K. Kotovsky, J.R. Hayes, and H.A. Simon. Why are some problems hard? Evidence from tower of Hanoi. *Cognitive psychology*, 17(2):248–294, 1985.
11. K. Kotovsky and H.A. Simon. What Makes Some Problems Really Hard: Explorations in the Problem Space of Difficulty. *Cognitive Psychology*, 22(2):143–83, 1990.
12. R. Pelánek. Difficulty rating of sudoku puzzles by a computational model. In *Proc. of Florida Artificial Intelligence Research Society Conference (FLAIRS 2011)*, 2011.
13. Z. Pizlo and Z. Li. Solving combinatorial problems: The 15-puzzle. *Memory and Cognition*, 33(6):1069, 2005.
14. H.A. Simon and A. Newell. *Human problem solving*. Prentice Hall, 1972.
15. W.J. van der Linden. A lognormal model for response times on test items. *Journal of Educational and Behavioral Statistics*, 31(2):181, 2006.
16. W.J. Van Der Linden. Conceptual issues in response-time modeling. *Journal of Educational Measurement*, 46(3):247–272, 2009.
17. K. Vanlehn. The behavior of tutoring systems. *International Journal of Artificial Intelligence in Education*, 16(3):227–265, 2006.