Int J Artif Intell Educ (2015) 25:493–519
DOI 10.1007/s40593-015-0048-x

ARTICLE

Student Modeling Based on Problem Solving Times

Radek Pelánek[1] · **Petr Jarušek**[1]

Published online: 13 June 2015

Abstract Student modeling in intelligent tutoring systems is mostly concerned with modeling correctness of students' answers. As interactive problem solving activities become increasingly common in educational systems, it is useful to focus also on timing information associated with problem solving. We argue that the focus on timing is natural for certain types of educational problems and we describe a simple model of problem solving times which assumes a linear relationship between a latent problem solving skill and a logarithm of a time to solve a problem. The model is closely related to models from two different areas: the item response theory and collaborative filtering. We describe two parameter estimation techniques for the model and several extensions – models with multidimensional skill, learning, or variability of performance. We describe an application of the proposed models in a widely used computerized practice system. Using both simulated data and real data from the system we evaluate the model, analyse its parameter values, and discuss the insight into problem difficulty which the model brings.

Keywords Intelligent tutoring systems · Student modeling · Problem solving times · Item response theory · Collaborative filtering · Computerized adaptive practice

✉ Radek Pelánek
pelanek@fi.muni.cz

[1] Faculty of Informatics, Masaryk University, Brno, Czech Republic

Introduction

Problem solving is an important part of education in general and of intelligent tutoring systems in particular. To use problem solving activities efficiently, it is important to estimate well their difficulty. Easy problems are boring, difficult problems are frustrating – this observation is elaborated in detail by the flow concept (Csikszentmihalyi 1991).

In intelligent tutoring systems (Anderson et al. 1985; Vanlehn 2006) student models typically focus on correctness of student answers (Desmarais and de Baker 2012), and correspondingly the problems in tutoring systems are designed mainly with the focus on correctness. This focus is partly due to historical and technical reasons – the easiest way to collect and evaluate student responses are multiple choice questions. Thanks to the advances in technology, however, it is now relatively easy to create rich interactive problem solving activities. In such environments it is useful to analyze not only correctness of students answers, but also timing information about the solution process.

As there exists extensive literature on modeling correctness of students' answers, in this work we focus solely on modeling problem solving times. To attain clear focus we study problems for which the timing information is sufficient measure of performance. A concrete, well-known example of such a problem is the Sudoku puzzle – for such a problem it is virtually impossible to guess the solution and thus time taken to reach the solution is a good measure of performance. A more educationally oriented example is a programming assignment, e.g., a task to write a program which outputs the first n primes. Time taken to write a program is a good indicator of student's programming skills, since debugging is an inherent part of programming. It is not important to get the program correct on the first attempt; the important thing is to be able to debug the program and make it work. In other domains (like mathematics), the focus on timing leads to development of new types of problems, which supplement and complement the traditional ones (more examples are discussed in Section "Examples of Problems").

Problems based on timing information also enable interesting immediate feedback to students, which helps to keep them in a flow state (Csikszentmihalyi 1991). Students can get information about comparison of their problem solving time with respect to other students (e.g., our system provides a histogram of times with the student's time highlighted). Such feedback leads to a natural competitiveness and supports motivation. Timing information is also useful for problem selection, sequencing, and recommendation. A tutoring system may for example start a session with a problem with small expected time to get students started quickly, or can use information about available time (e.g., during school class) to select a suitable set of problems (Michlík and Bieliková 2010).

Standard student models (with focus on correctness of answers) specify a probability that a student will answer correctly. In our setting, models specify a time in which a student will solve a problem, or more exactly a distribution of time. We propose a model which assumes a linear relationship between a problem solving skill and a logarithm of time to solve a problem, i.e., exponential relation between skill and time. We also propose several extensions of the model (modeling variability of

students' performance, learning, and multidimensional skill). We describe parameter estimation procedures based on maximum likelihood. The described models have close connections to two different areas – the item response theory and collaborative filtering. Analogically to models in item response theory, the described models have a group invariance property, i.e., the estimated problem parameters are not influenced by a group of students which solve the problem. This is important since difficult problems are typically not solved by a random sample of students, but rather only by above average students.

The proposed models are generative and thus we can perform experiments with simulated data. These experiments provide insight into how much data are needed in order to get usable estimates of parameter values. Moreover, the model is currently used in a "Problem Solving Tutor" – a web-based system which recommends students problems of suitable difficulty. The tutor contains 30 types of problems (more than 1400 specific problem instances) mainly from areas of programming, mathematics, and logic puzzles. The system is used in several schools and contains data from more than 10 000 users and 400 000 solved problems. Using this data we evaluate the model and its different variants.

The evaluation shows several interesting results. The data support the basic model assumption of linear relationship between skill and a logarithm of time to solve a problem. For predicting future times even a simple baseline predictor provides reasonable results; the model achieves only a slight improvement in predictions. The extension with multidimensional skill can be used to automatically classify problems into categories. The contribution of the model is mainly in the insight into problem and student characteristics – we can determine not just average difficulty of problems, but also discrimination of problems or the degree of student learning. The model parameters can be used to guide the behaviour of the tutoring system and also as a feedback for both students and content authors.

Related Work

Our work is related to four research areas, but has significant differences from each of them. We propose a novel model of problem solving times, which is a variation of models used in the *item response theory* and *collaborative filtering*. The model is used in a *tutoring system* for practising *problem solving*. The system *recommends* users problems of suitable difficulty. The paper summarizes and extends the preliminary results reported in Jarušek and Pelánek (2011a, 2012a, b), Jarušek et al. (2013), Borŏs et al. (2013) and Jarušek (2013).

Item Response Theory

The item response theory (IRT) (Baker 2001; De Ayala 2008) is used particularly for computerized adaptive testing. IRT considers tests where each question (item) has several possible answers. IRT models specify relationship between a latent student skill 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 time to solve a problem. IRT is also primarily applied for adaptive testing and thus strongly uses the assumption of a constant skill – the skill is not supposed to change during a test. We are interested in intelligent tutoring and adaptive practice, i.e., not only measuring students' skill, but also improving it. Thus we also want to be able to model changes in skill.

The basic models of IRT assume that probability of correct response depends on one latent skill θ. The most often used model is the three parameter logistic model $p(correct|a, b, c, \theta) = c + (1 - c)e^{a(\theta-b)}/(1 + e^{a(\theta-b)})$. This model has three parameters (see Fig. 1): b is the basic difficulty of an item, a is the discrimination factor (slope of the curve, how well the item discriminates based on skill), and c is the pseudo-guessing parameter (lower limit of the curve, probability that even a student with very low skill will guess the correct answer). Our model is similar, but instead of modeling probability of correct answer, we model a distribution of a problem solving times. We have intentionally chosen notation for our model to be analogous to the IRT three parameter model, the analogy is illustrated in Fig. 1.

The most relevant aspect of IRT are models of response times (Van Der Linden 2006, 2009). The research in IRT uses the timing information as a secondary source of data on students skills or combines the correctness and timing information, e.g., in the form of a hierarchical model (Van Der Linden 2009). The timing information is sometimes also used for item selection (Fan et al. 2012). We focus solely on timing information and thus we are able to model it in more detail, e.g., IRT models do not consider different variance of times for items and students or changes in skill.

Intelligent Tutoring Systems and Student Modeling

Intelligent tutoring systems (Anderson et al. 1985) 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 learning algebra (Koedinger et al. 1997; Koedinger et al. 2000). Tutoring systems usually have static structure

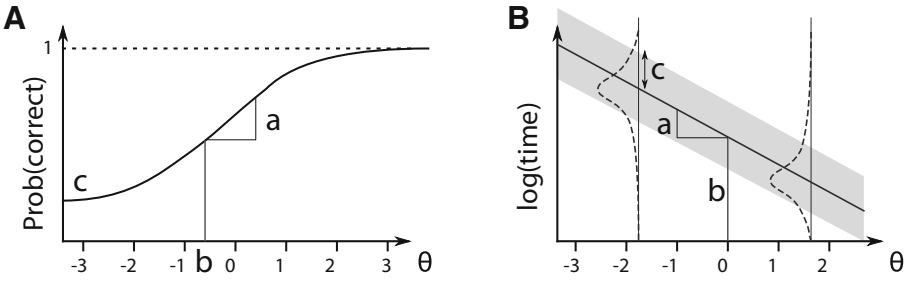

Fig. 1 An intuitive illustration of the analogy between the IRT three parameter model (**A**) and our model (**B**). *Dashed lines* illustrate distributions for certain skill; *solid line* denotes the expected problem solving time, *grey area* depicts the area into which most attempts should fall

determined by an expert in a particular domain. Our system is dynamic and recommends problems based on collected problem solving data. We focus solely on the "outer loop" of tutoring (Vanlehn 2006), i.e., how to dynamically order problems.

Intelligent tutoring systems use student models to assess the knowledge of students. Similarly to the situation in IRT, student models use mainly data on correctness of answers. For example, recent review of learner models (Desmarais and de Baker 2012) describes only models of this type and does not consider timing information at all. Even if the tool provides interactive problem solving setting, data that are collected and modeled are mostly of the test question type (correct/incorrect). In this work we take the other approach. We focus solely on the timing information. We consider well-structured problems with clearly defined correct solution and use a problem solving time as the single measure of student performance. There are no other "quality of solution" measures, i.e., no hints during solutions or acceptance of partial solutions. Students must solve a problem until they reach a correct solution (or they can abandon the problem and try a simpler one).

Some researchers have considered timing information for student modeling, but usually as a supplement to the correctness data. Response times have been used in an extension of knowledge tracing (Wang and Heffernan 2012). Reading times are used together with correctness information in the LISTEN project (Mostow and Aist 2001; Beck and Mostow 2008). In other applications, response times are usually used as one of many studied features (Beck and Woolf 2000), e.g., for causal modeling of educational data (Rai and Beck 2011) or as a proxy for self-explanation (Conati et al. 2002). Another supplementary use of timing information is to model affective states of students, e.g., effort (Arroyo et al. 2010), or disengagement (Beck 2004). In the case of systems for practicing factual knowledge (e.g., vocabulary), spacing and forgetting effects are important (Pavlik and Anderson 2005). Models for this domain take into account timing information, particularly the time between attempts (Pavlik and Anderson 2008; Pavlik et al. 2008).

Recommender Systems and Collaborative Filtering

Recommender systems (Kantor et al. 2010) 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. Our system works in the same way, although we are not interested in recommending products, but problems of suitable difficulty. Otherwise our situation is analogical – in both cases the input is a large sparse matrix, the outputs are predictions of missing values. Collaborative filtering techniques have been used previously in the context of item response theory (Bergner et al. 2012) and tutoring systems (Thai-Nghe et al. 2011) for predictions of correctness of students' answers.

One of the approaches to collaborative filtering is based on matrix factorization methods, particularly on the singular value decomposition. It leads to the following

model (Koren and Bell 2011): $r_{ui} = b_i + b_u + \mathbf{q_i}^T \cdot \mathbf{p_u} + \epsilon$, where u is an user, i is an item, r_{ui} is the predicted rating, $\mathbf{q_i}$ and $\mathbf{p_u}$ are vectors of length k specifying problem-feature and user-feature interactions, b_i and b_u are item and user biases, and ϵ is a random noise. The parameters of the model are typically estimated using stochastic gradient descent with the goal to minimize sum of square errors. Our basic model is similar to this approach for $k = 1$, where users are interpreted as students, items as problems, and ratings as problem solving times.

Human Problem Solving

Research in human problem solving (Simon and Newell 1972) so far focused mainly on analysis and computational modeling of human problem solving for a particular problem, mostly in the domain of logic puzzles e.g., Tower of Hanoi (Kotovsky et al. 1985), Chinese ring puzzle (Kotovsky and Simon 1990), Fifteen puzzle (Pizlo and Li 2005), Sokoban (Jarušek and Pelánek 2011b), or Sudoku (Pelánek 2011). This research provides insight into problem difficulty and problem solving times, but the insight is usually closely tied to a particular problem. In this work we use a "black box" approach – we use for the modeling only the overall problem solving time and the model is thus independent of the specifics of a particular problem. Since our models have the "cold start problem", computational models of human problem solving (when available) can be used for setting initial estimates of problem parameters before data are available.

Modeling Problem Solving Times

We describe the general modeling approach, a specific model, and several extensions of the basic model.

General Approach

We assume that we have a set of students S, a set of problems P, and data on problem solving times: t_{sp} is a logarithm of a time it took a student $s \in S$ to solve a problem $p \in P$. We consider only times for successfully completed problems, i.e., T is a matrix with missing values as some of the students did not solve (or even attempt) all problems. In this work we do not consider any other information about students and problems except for the problem solving times. We study models for predicting future problem solving times based on the available data.

As noted above, we work with a logarithm of time instead of the untransformed time itself. There are several good reasons to do so. At first, problem solving times have a natural "multiplicative" (not "additive") nature, e.g., if Alice is a slightly better problem solver than Bob, then we expect her times to be 0.8 of Bob's times (not 20 second smaller than Bob's times). At second, previous research on response times in item response theory successfully used the assumption of log-normal distribution of response times (see e.g., Van Der Linden 2006, 2009) and analysis of our data also suggests that problem solving times are log-normally distributed. At

third, the use of a logarithm of time has both pragmatic advantages (e.g., reduction of effect of outliers) and theoretical advantages. Particularly, given the use of a logarithm of time, the data can be well modeled by simple linear models with Gaussian noise.

The Basic Model

The basic model that we propose assumes that a problem solving performance depends on one latent problem solving skill θ and three main problem parameters: a basic difficulty of the problem b, a discrimination factor a, and a randomness factor c. The basic structure of the model is simple – a linear model with Gaussian noise:

$$t = b + a\theta + \epsilon \qquad \epsilon = \mathcal{N}(0, c^2)$$

Basic difficulty b describes expected solving time of a student with average skill. Discrimination factor a describes the slope of the function, i.e., it specifies how the problem distinguishes between students with different skills. The randomness factor c describes the variability of problem solving times for students with the same skill. The model is relatively simple, yet it can capture different aspects of problem difficulty and their combinations (see Fig. 2).

The model is analogical to models used in item response theory for modeling probability of a correct answer, our notation was chosen to stress this analogy (see Fig. 1 in Related work). In the following we use subscript s to index student parameters and subscript p to index problem parameters, i.e., the model is written as $t_{sp} = b_p + a_p\theta_s + \mathcal{N}(0, c_s^2)$.

Normalization and Group Invariance

Analogically to item response theory models (De Ayala 2008), our model suffers from the "indeterminacy of the scale" and has the "group invariance" feature. The "indeterminacy of the scale" means that the model is not well identified, e.g., if we multiply all skills by 2 and divide all discrimination factors by 2, the model predictions remain the same. In some students models identifiability is an important

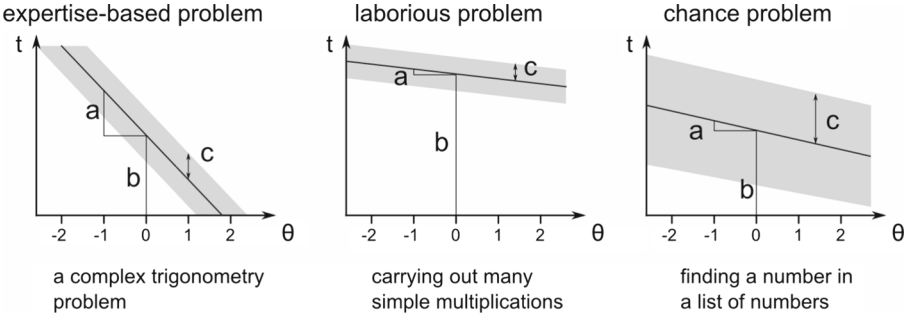

Fig. 2 Three different problem types as captured by the basic model. The *bottom row* provides specific examples of problems of a particular type

problem, e.g., Beck and Chang (2007) discuss the problem in the case of Bayesian knowledge tracing. In our case the issue can be solved simply by normalization – we require that the mean of all θ_s is 0 and the mean of all a_p is -1. This ensures that skills have clear and natural interpretation.

"Group invariance" is an important feature of the model. It means that the problem (resp. student) parameters do not depend on a subgroup of students which solved the problem (resp. problems solved by a student). Let us explain this important feature by comparing our model with a mean time to solve a problem, which is a typical metric of problem difficulty. In both cases the basic 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 often happens that harder problems are solved only by students with above-average skill (this is certainly true in our system). In this case the mean time underestimates the real difficulty of the problem, whereas our model is not biased by the selection of students. Figure 3a provides a specific illustration: a problem solved by a group of above average students, which leads to the difficulty parameter b being higher than the mean time. If the same problem was solved by a group of below average students, the difficulty parameter b would remain approximately the same, whereas the mean time would significantly increase.

There remains a potential bias caused by the fact that we consider only successful attempts. If some students solve only problems where they can see solution quickly and abandon others, their skill estimates may be inflated. If a problem has a large number of unsuccessful solvers who spent lot of time trying to solve it, it is probably more difficult than the analysis of successful attempts suggests. For our data, however, such cases do not occur in any significant way; inclusion of unsuccessful attempts did not bring any improvement of models. Therefore, we do not discuss unsuccessful attempts in the discussion of our models, but we acknowledge that in different contexts this information may be worth considering.

Modeling Variability of Students' Performance

The basic model outlined above assumes that the noise depends only on the characteristics of a problem. But some students are more consistent in their performance

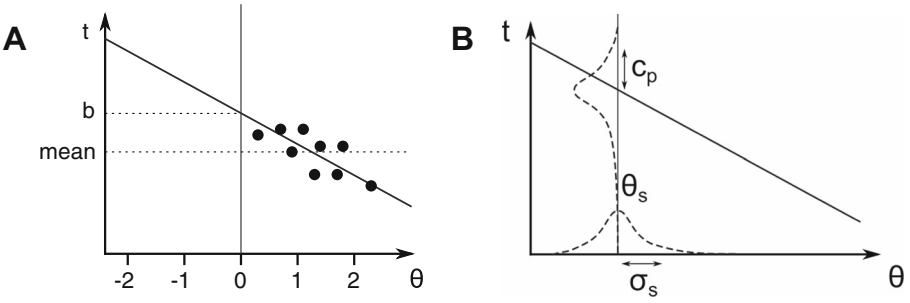

Fig. 3 **A** If a problem is solved by above-average students, the mean time underestimates the difficulty of a problem, whereas our model can capture it correctly. **B** Model with variability of students' performance

than others and thus it is sensible to add a dependence of the noise on students. To incorporate variability of students' performance we add a new student parameter σ and assume that the noise is given as $\mathcal{N}(0, c_p^2 + \sigma_s^2 a_p^2)$ – the variance is a weighted sum of a problem variance c_p^2 and a student variance σ_s^2, where student's contribution to the variance depends on the discrimination of the problem. Intuitively, student's characteristics matter particularly for more discriminating problems.

The model with variance $c^2 + a^2\sigma^2$ is equivalent to the following approach: "at first determine a student's local skill for the attempt (based on his variance σ^2) and then determine the problem solving time with respect to this local skill" (Fig. 3b):

$$t_{sp} = b_p + a_p\theta_l + \epsilon \qquad \epsilon = \mathcal{N}(0, c^2) \qquad \theta_l = \mathcal{N}(\theta_s, \sigma_s^2)$$

The equivalence of the two definitions is a special case of a general result for Gaussian distributions (see e.g., Bishop 2006).

Modeling Learning

Another sensible extension is to incorporate learning into the model. The basic model assumes a fixed problem solving skill, but students' skill should improve as they solve more problems – that is, after all, the aim of tutoring systems. The model extension is inspired by the extensive research on learning curves.

A learning curve is a graph which plots the performance on a task (usually time or error rate) versus the number of trials. The shape of the learning curve is in the majority of human activities driven by a power law: $T = BN^{-\alpha}$, where T is the predicted time, N the number of trials, α the learning rate and B the performance at the first trial. Other curves like hyperbolic, logistic or exponential were tested as well (Newell and Rosenbloom 1981), but seem to do better only in a few cases. Learning curves are often examined in psychology (LaBerge 1975) to capture cognitive skills or memory processes as well as in economy (Hirsch 1952). The slope and fit of the curve is measured to find out the quality of learning or progress taking place. Finally, these curves are used to compare students, tasks or methods in order to improve learning process, tutor systems (Martin et al. 2011) or business strategies.

If we take the logarithm of the above mentioned form of the power law, it can be naturally combined with our basic model of problem solving times:

$$t_{sp} = b_p + a_p(\theta_s + \delta_s \cdot \log(k_{sp})) + \epsilon$$

where δ_s is a student's learning rate and k_{sp} is the order of a problem p in a problem solving sequence of a student s. In the current analysis of this model we assume constant variance. Nevertheless, the model can be easily combined with the more detailed model of the noise presented above.

Model with Multidimensional Skill

Another simplifying assumption of the basic model is the assumption of a one dimensional skills. This assumption is reasonable for simple problems like logic puzzles, but for more complex educational problems (like the function exercise) it may be too

simplifying to assume that the students ability to solve problems can be captured by a one dimensional variable.

Thus we consider an extension of the model with a k dimensional skills: $t_{sp} = b_p + \mathbf{a_p}^T \boldsymbol{\theta}_\mathbf{s} + \epsilon$, where $\mathbf{a_p}$ is a k dimensional discrimination vector and $\boldsymbol{\theta}_\mathbf{s}$ is a k dimensional skill vector. We can also view the model using matrix multiplication, with A being a $|P| \times k$ matrix mapping problems to concepts, and Θ a $|S| \times k$ matrix mapping students to concepts. The model can be then written as $T = \mathbf{b} + \Theta A^T + E$, where E is a Gaussian noise matrix. The model is thus analogical to widely used Q-matrix models (Tatsuoka 1983). The main difference is that standard Q-matrices are typically used in the setting of test questions with binary response (0 – incorrect, 1 – correct), with the Q-matrix entries and student skills being also binary (the model specifies probability of a correct answer using noisy and/or function). For this binary setting, there has been extensive research, e.g., into Q-matrix mining from the data (Barnes 2005; Barnes et al. 2005; Desmarais et al. 2012), or validation of expert provided matrix (De La Torre 2008; Rupp and Templin 2008). In our setting it is natural to allow both Q-matrix values (discrimination factors) and skills to be continuous.

The fitted model can be used for discrete classification of problems. For example in our experiments (Section "Detection of Concepts") we perform classification into two classes. In this case we use a model with 2 concepts, fit the values a_{p1} and a_{p2}, and classify a problem p by comparing these two values.

Parameter Estimation

Table 1 gives summary of presented models. Now we will discuss how to estimate parameters of these models. Note that for the model with learning and multidimensional skill we assume a constant noise (in order to simplify the estimation of parameters).

To estimate the model parameters we use the maximum likelihood approach, i.e., we are trying to find the values of parameters which maximize the likelihood of the observed data. As it is intractable to find such values analytically, we consider two approximative approaches: iterative computation of joint maximum likelihood and stochastic gradient descent. Both these techniques are commonly used in item

Table 1 Overview of proposed models

Model variant	Model	Noise ϵ
Basic model	$b_p + a_p\theta_s + \epsilon$	$\mathcal{N}(0, c_p)$
Var. of students' performance	$b_p + a_p\theta_s + \epsilon$	$\mathcal{N}(0, c_p^2 + a_p^2\sigma_s^2)$
Model with learning	$b_p + a_p(\theta_s + \delta_s \cdot \log(k_{sp})) + \epsilon$	$\mathcal{N}(0, k)$
Multidimensional skill	$b_p + \mathbf{a_p}^T \cdot \boldsymbol{\theta}_\mathbf{s} + \epsilon$	$\mathcal{N}(0, k)$

response theory and machine learning and thus we focus mainly on issues specific for our models. A more detailed explanation can be found in Jarušek (2013).

In the description of parameter estimation procedures we use the following notation: $Solved$ is a set of tuples (s, p) such that a student s solved a problem p (i.e, t_{sp} is known), P_s is a set of problems solved by a student s, and S_p is a set of students who solved problem p. We describe the proposed parameter estimation procedures in the form of pseudocode; the implementation was done in Python.

Iterative Computation of Joint Maximum Likelihood

The first method is the iterative computation of joint maximum likelihood, which is often used in the item response theory. We describe the approach for the basic model. Problem parameters a_p, b_p, c_p and student skills θ_s are estimated using an iterative computation: problem parameters are computed using estimates of student skills; student skills are improved using estimates of problem parameters (both directions are computed by maximum likelihood estimation); and this process continues until requested precision is reached.

Suppose that we know problem parameters a_p, b_p, c_p of all problems and based on the data we want to estimate a skill θ_s of a particular student. The likelihood of the observed times t_{sp} given our basic model is:

$$L = \prod_{p \in P_s} \mathcal{N}(b_p + a_p\theta_s, c_p^2)(t_{sp}) = k \prod_{p \in P_s} \frac{1}{c_p} e^{-\frac{(t_{sp}-(b_p+a_p\theta_s))^2}{2c_p^2}}$$

We need to find the value of θ_s such that L is maximized. As is usual, we proceed by finding maximum of $\ln L$, which is the same as maximum of L. Since $\ln L$ is a quadratic function in θ_s, we can find maximum by finding the value of θ_s for which the derivation is zero. The solution is:

$$\theta_s = \frac{\sum_{p \in P_s} \frac{a_p^2}{c_p^2} \frac{t_{sp}-b_p}{a_p}}{\sum_{p \in P_s} \frac{a_p^2}{c_p^2}}$$

The resulting expression for θ_s has a clear intuitive interpretation. The expression $(t_{sp} - b_p)/a_p$ is a "local estimate of skill" for the problem p, i.e., it is the value of θ_s for which the expected time is t_{sp}. The overall estimate of θ_s is obtained as a weighted average of these local estimates, where the weight is given by the expression a_p^2/c_p^2, i.e., the more discriminating and less noisy a problem is, the more weight it gets.

For the other direction we need to estimate problem parameters a_p, b_p, c_p given student skills θ_s. In this case maximal likelihood estimates can be found by a regression analysis. For the basic model 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.

The joint estimate of parameters is computed by an iterative process, which is described in Algorithm 1 in the form of pseudocode. At first we initialize the problem parameters and then we repeatedly use the above described estimation of skills and estimation of problem parameters until a chosen convergence criterion is satisfied (in practice the convergence is very fast and 3 iterations are sufficient).

Algorithm 1 Parameter estimation using iterative computation of maximum likelihood estimates.

for *all problems* p **do**
  $a_p := -1$
  $b_p := \text{mean}(\{t_{sp} | s \in S_p\})$
  $c_p := 1$
repeat
  for *all students* s **do**
    $\theta_s := (\sum_{p \in P_s} \frac{a_p^2}{c_p^2} \frac{t_{sp} - b_p}{a_p}) / (\sum_{p \in P_s} \frac{a_p^2}{c_p^2})$
  for *all problems* p **do**
    $a_p, b_p :=$ linear regression$(\theta, \mathbf{t_{\cdot p}})$
    $c_p :=$ standard deviation (residuals from linear regression)
until *convergence*

Stochastic Gradient Descent

In the iterative computation of joint maximum likelihood approach we used closed form expression for both directions. This may be difficult for the model extensions. Thus we also consider the stochastic gradient descent, which estimates problem and student parameters at the same time using greedy improvement. This techniques is often used in collaborative filtering. We describe details of the approach for the model with variance, other cases are analogical (usually simpler).

The application of stochastic gradient describe for maximum likelihood estimation is rather standard approach (see e.g., Bishop 2006), therefore we focus in the following description only on the derivation of the error function and the gradient. The likelihood of observed times t_{sp} is:

$$L = \prod_{s,p} \mathcal{N}\left(b_p + a_p\theta_s, c_p^2 + a_p^2\sigma_s^2\right)(t_{sp})$$

To make the derivation more readable, we introduce the following notation:

- $e_{sp} = t_{sp} - (b_p + a_p\theta_s)$ (prediction error for a student and a problem),
- $v_{sp} = c_p^2 + a_p^2\sigma_s^2$ (variance for a student and a problem).

Maximizing the likelihood is equivalent to minimizing the error function $E = \sum_{s,p} E_{sp}$, where $E_{sp} = \frac{1}{2}(\frac{e_{sp}^2}{v_{sp}} + \ln(v_{sp}))$. It is intractable to find the minimum

analytically, but we can minimize the function using stochastic gradient descent. To do so we need to compute a gradient of E_{sp}:

$$\begin{aligned}
\frac{\partial E_{sp}}{\partial a_p} &= \frac{-e_{sp}\theta_s v_{sp} - a_p\sigma_s^2 e_{sp}^2}{v_{sp}^2} + \frac{a_p\sigma_s^2}{v_{sp}} = -\frac{e_{sp}}{v_{sp}}\left(\theta_s + a_p\sigma_s^2\frac{e_{sp}}{v_{sp}}\right) + \frac{a_p\sigma_s^2}{v_{sp}} \\
\frac{\partial E_{sp}}{\partial b_p} &= -\frac{e_{sp}}{v_{sp}} \\
\frac{\partial E_{sp}}{\partial \theta_s} &= -a_p\frac{e_{sp}}{v_{sp}} \\
\frac{\partial E_{sp}}{\partial c_p^2} &= \frac{1}{2}\left(-\frac{e_{sp}^2}{v_{sp}^2} + \frac{1}{v_{sp}}\right) = -\frac{1}{2v_{sp}^2}(e_{sp}^2 - v_{sp}) \\
\frac{\partial E_{sp}}{\partial \sigma_s^2} &= \frac{1}{2}\left(-a^2\frac{e_{sp}^2}{v_{sp}^2} + a^2\frac{1}{v_{sp}}\right) = -\frac{a^2}{2v_{sp}^2}(e_{sp}^2 - v_{sp})
\end{aligned}$$

As with the previous expression, the results have in most cases straightforward intuitive interpretation. For example the gradient with respect to θ_s is $-a_p\frac{e_{sp}}{v_{sp}}$, i.e., the estimation procedure gives more weight to attempts over problems which are more discriminating and have smaller variance.

Stochastic gradient descent can find only local minima. However, by good initialization we can improve the chance of finding a global optimum. In our case it is possible to get a good initial estimate of parameters quite easily. Algorithm 2 specifies such initialization and provides summary of the whole procedure in the form of pseudocode. The parameter update rule uses a meta-parameter γ (usually called step size or learning rate), we used $\gamma = 0.005$, which leads to a good trade-off between the stability and speed of convergence of gradient descent (the same value has been used in previous applications of gradient descent in collaborative filtering Koren and Bell 2011).

If we make a simplifying assumption and assume that the variance is constant (independent of a particular problem and student), then the error function is the basic sum-of-squares error function and the computation of gradient simplifies to: $\frac{\partial E_{sp}}{\partial a_p} = -\theta_s e_{sp}$, $\frac{\partial E_{sp}}{\partial b_p} = -e_{sp}$, $\frac{\partial E_{sp}}{\partial \theta_s} = -a_p e_{sp}$.

The method can be easily modified for estimation of parameters for model with learning. For the model with multidimensional skill the main complication with respect to the one dimensional case is the initialization of the gradient descent. In the one dimensional case it is easy to obtain a good initial estimates of parameters, and thus the gradient descent is stable. In the multidimensional case if we do not use any additional information about problems or students, there is no reasonable way to initialize discrimination factors $\mathbf{a_p}$ and students skills $\boldsymbol{\theta_s}$. At least we have to break the symmetry of individual skills, so we use a randomized initialization. The gradient descent is thus less stable than in the one dimensional case and it is necessary to run multiple runs in order to find a good local minimum. Note that the approach is analogical to Q-matrix method (Barnes 2005) in the discrete setting (multiple random initializations followed by a greedy improvement).

Algorithm 2 Parameter estimation using stochastic gradient descent.

for *all problems* p **do**
  $a_p := -1$
  $b_p := \text{mean}(\{t_{sp} | s \in S_p\})$
  $c_p := \frac{1}{2}\ \text{variance}(\{b_p - t_{sp} | s \in S_p\})$
for *all students* s **do**
  $\theta_s := \text{mean}(\{b_p - t_{sp} | p \in P_s\})$
  $\sigma_s^2 := \frac{1}{2}\ \text{variance}(\{b_p - t_{sp} | p \in P_s\})$
repeat
  for *all* $(s, p) \in$ *Solved in randomized order* **do**
    $e_{sp} := t_{sp} - (b_p + a_p \theta_s)$
    $v_{sp} = c_p^2 + a_p^2 \sigma_s^2$
    for *all parameters* x *in* $[a_p, b_p, c_p, \sigma_p^2, \theta_s]$ *in parallel* **do**
      $x := x - \gamma \cdot \frac{\partial E_{sp}}{\partial x}$
until *convergence*

Application: Problem Solving Tutor

The described models are currently used in our Problem Solving Tutor – a free web-based tutoring system for practicing problem solving (available at tutor.fi.muni.cz). At the moment the system focuses solely on the "outer loop" of intelligent tutoring (Vanlehn 2006), i.e., the system can be described as providing "computerized adaptive practice" by recommending problem instances of the right difficulty (a similar system is for example the Maths Garden system; Klinkenberg et al. 2011).

Functionality of the System

The system adapts to an individual student – based on past problem solving attempts the system estimates student's problem solving skill, using this estimated skill it predicts problem solving times for new problems and chooses a suitable problem for a student.

Predicted problem solving time is used for recommendations of problems to solve. When a student finishes a problem, the system recommends two new problems to solve – one of higher difficulty, one of lower difficulty; the change of difficulty is specified as a percentual difference with respect to the expected problem solving time for the last solved problem. The selection of problems is done using a linear scoring function, which combines several problem attributes: expected problem solving time, problem randomness (preferring more deterministic problems), and information whether the student has previously attempted to solve the problem. For specific weights of these attributes see Jarušek (2013).

For each unsolved problem a student can see a personalized prediction of problem solving time. After solving the problem, the system provides comparison of the

achieved time with performance of other student as well as with the personalized prediction. This use of personalized predictions has a side-effect of adding a game-like element into the system, since students can compete with their predicted problem solving time. This predicted time is "by construction" a suitable challenge for them, supporting a state of flow (Csikszentmihalyi 1991).

The system also contains support for virtual classes and thus can be easily used in a classroom (Jarušek 2013). This aspect of the system is similar to many other tutoring systems and thus we do not elaborate on it.

Gradual Start

Similarly to other recommendation systems which make recommendations based on collected data, we have to face the "cold start" problem: How to make recommendations when we do not have enough data? We use a "gradual start" approach. At the beginning we provide estimates of solving times of individual problems and add into the system several artificial users with these estimated problem solving times. The estimates can be obtained from problem authors, by naive metrics (like the length of the shortest path to a solution), or by using computational models of human problem solvers (Jarušek and Pelánek 2011b; Pelánek 2011).

At the beginning the system uses a simplified model, in which parameters a, c are fixed, i.e., the only problem parameter used is the basic difficulty b. This simplified model is more robust to random noise in data (this noise is particularly significant with sparse data). As the system collects more data, we still make predictions using the simplified model, but at the same time we compute more complex models and evaluate their predictions. As soon as predictions of more complex models become better, we start using them for displaying predictions and recommendations.

Use of Problem Parameters

The problem parameters (difficulty, discrimination, randomness) are used as a feedback for system developers and content authors. This feedback is valuable for getting insight into pedagogical properties of individual problems (see Section "Insight Gained from Parameter Values") and is also useful for detecting problems behaving in unexpected ways, e.g., problems with different solutions than the author intended. Analysis of model parameters (particularly discrepancies in values of the discrimination parameter) also enabled us to detect cheating by a some students.[1] This observation led to a detailed analysis of cheating within the project and development of methods for automatic detection of cheating by anomaly detection techniques (Šormová 2014).

[1]The cheating was enabled by a technical weakness in the JavaScript implementation of one of the problems, where it was possible to find the problem solution within the HTML source of the webpage.

Examples of Problems

We describe several specific problems to give the reader a better idea of the kind of problems we are considering and of the comparison to the usual type of problems and models:

- *Graphs and functions.* The goal is to identify a formula for describing a function specified by its graph. Students may use an interactive graph drawing facility which plots their attempts. Incorrect attempts are not penalized, students try different functions until they find a solution.
- *Linear transformations.* The goal is to transform a given shape into a target shape using linear transformation (rotation, scaling, reflexion). Transformations are specified using either buttons for individual transformations or using a matrix notation.
- *Robot programming.* Introductory programming exercise with a virtual robot. Robot is programmed via simple commands, e.g., move forward, turn left, repeat, conditional execution. Programs are specified graphically, the task is to program a robot to solve a given problem (e.g., collect all flowers in a maze).
- *Standard programming.* The goal is to write a program for a specified task, programs are tested over (hidden) testing data. If a submitted program is incorrect, students are provided with specific inputs on which the program does not work correctly.
- *Finite automata.* The goal is to construct a finite automaton accepting a language specified by a list of words, a regular expression, or a regular grammar. The interface is interactive and can automatically provide counterexamples for incorrect automata.
- *Logic puzzles.* Well-structured logic puzzles with clearly defined rules and goals, e.g., well-known puzzles like Sudoku, Nurikabe, Sokoban, Rush hour.

In all cases the problems are well-structured and the system can algorithmically recognize a correct solution. Problems are interactive and students get immediate feedback on individual attempts. Note that in all cases it is impossible to just guess the solution without thinking, i.e., problem solving time is a good measure of performance.

The problem solving formulation leads to iterative approach to solution and helps to build intuition about the problem domain. Moreover, when we are able to find a suitable problem solving formulation, students often find problems quite attractive and are willing to do activities that would otherwise be considered rather boring (like practicing functions).

Available Data

The system contains 30 types of problems, particularly computer science problems, math problems, and logic puzzles (specific examples are described in Section "Examples of Problems"). Together Problem Solving Tutor contains more than 1400 problem instances. Most of the problems are accessible in English, some are so far implemented only in Czech.

The system is already used by more than 20 schools and has more than 10 000 registered users (mainly university and high school students). Users have spent more then 13 000 hours solving more than 400 000 problems. The collected data are used for the below described evaluation. The number of solved problems is distributed unevenly among different problem types, in the evaluation we use only the 8 most solved problems.

Evaluation

Now we report on an evaluation of the parameter estimation procedure, model prediction, parameter values, and other experiments with the model. The experiments were performed using both simulated data and data about real students from the Problem Solving Tutor (described in Section "Application: Problem Solving Tutor"). We also give a specific example of an insight into problem difficulty which the model brings.

For the experiment we used 8 most solved problem types from the Problem Solving Tutor, the basic statistics about these problems are given in Table 2. For each problem we consider only students who solved at least 15 instances of this problem. Note that the problems used for evaluation are mostly logic puzzles as we have most data for these problems. The Problem Solving Tutor also contains educational problems from mathematics and computer science, the results for these problems are similar to the reported results (only less robust due to the smaller extent of the data).

In previous section we described two parameter estimation procedures. For the basic model the estimated parameter values are nearly identical and the choice of the parameter estimation procedure does not influence the reported results. Advantage of iterative computation of joint maximum likelihood is speed (gradient descent needs more iterations to converge), advantage of the gradient descent is its direct applicability to extended models.

The model assumes normal errors under the log-linear model. To test this assumption we performed an analysis of residuals. The residuals fit the normal distribution well, usually there is a very slight positive skew (see Fig. 4 which shows results for

Table 2 Data used for evaluation

Problem type	Students	Problem instances	Solved problems
Tilt Maze	2091	110	43544
Robotanist	1254	68	30467
Binary crossword	778	57	23983
Region puzzle	313	112	14113
Slitherlink	204	88	10264
Sokoban	294	69	9471
Rush Hour	1092	69	9471
Nurikabe	132	46	4665

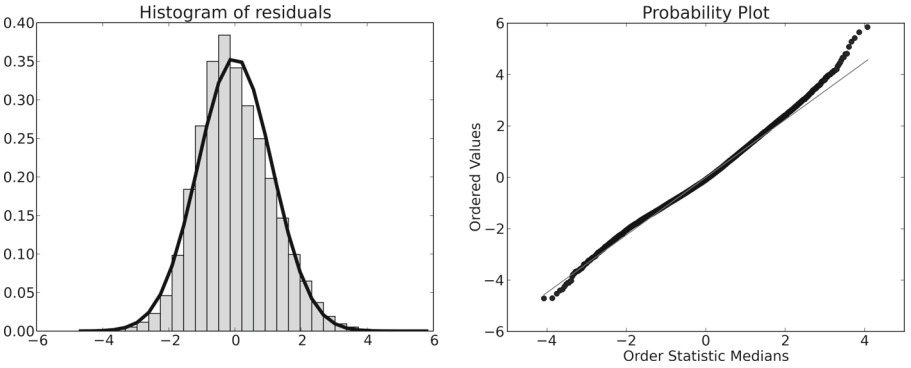

Fig. 4 Analysis of residuals for the Robotanist problem

one of the problems). This skew is probably caused by "interrupted solution attempts" rather than by an inherent violation of the assumptions about human problem solving behaviour. Thus it may be useful to filter the data using for example anomaly detection techniques (Chandola et al. 2009; Šormová 2014).

Simulated Data: Model with Variance

The described models can be easily used to generate simulated data. Even though we have large scale data on real students, the simulated data are still useful, because for these data we know "correct answers" and thus we can thoroughly evaluate the parameter estimation procedure.

To generate data we have to specify the "meta-parameters" of student and problem populations (distributions of student skills and problem parameters). We have specified these meta-parameters in such a way that the simulated data are similar to the data from real students and problems from the Problem Solving Tutor, e.g., the student skill θ is distributed normally with mean 0 and standard deviation 0.7, problem parameter b is distributed normally with mean 7 and standard deviation 2. The results are reported for simulated data that contain values for all student-problem combinations (real data contain missing values). Nevertheless, the results are very similar even with missing values.

Using such simulated data we can get an insight into how well the parameter estimation procedure determines values of parameters and how many students (problems) are needed to get a good estimate of parameter values. Figure 5 shows results for the model with variable student performance for varying number of students (left) and varying number of problems (right). The graph shows the Spearman's correlation coefficient between the estimated values of parameters and their true values; results are averaged over 10 runs. We have chosen Spearman's correlation coefficient as a metric for this evaluation because we are typically interested more in relative values of parameters than in their absolute values – the values are used to sort problems (based on their difficulty or discrimination) or students (based on their skill).

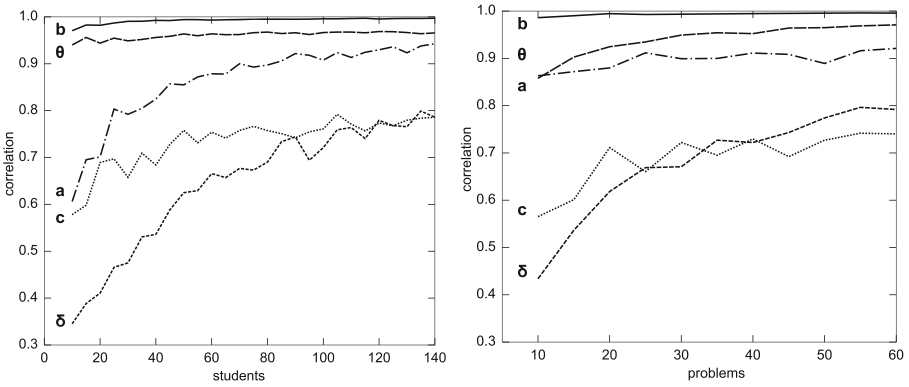

Fig. 5 Results for simulated data: Spearman's correlation coefficient for true values and computed values of problem parameters a, b, c and student parameters θ, σ. *Left*: fixed number of problems (50) and varying number of students. *Right*: fixed number of students (100) and varying number of problems

The results show that the basic difficulty of problems b and students' skill θ can be estimated easily even from relatively few data. Estimating problem discrimination a is more difficult – to get a good estimate we need data from at least 30 solvers and even with more data the further improvement of the estimates is slow. As could be expected, it is most difficult to get a reasonable estimate of student and problem variance. To do so we need data from at least 50 problems and 150 students.

Simulated Data: Model with Learning

Now we investigate model with learning, i.e., how well we are able to detect students' learning rates and differentiate between students who are learning and who are not. The basic observation here is straightforward: If the differences in learning rates are high and noise is low, then it is easy to detect the learning in the data. If the students' learning rates are very similar and noise in data is high, it is impossible to detect the learning. The graph in Fig. 6 shows the transition between these two extremes.

In many practical cases the ordering, in which students solve problems, is very similar. Often students proceed from simpler problems to more difficult ones – this is certainly true for our data, which are used in the next section. Does this correlated ordering influence the estimation of parameters from data?

Consider the extreme case when all students solve the problem in the same order. Then the model is not well identified. If we increase the values of all student learning rates δ_s by x and decrease the values of all problem parameters b_p by $xa_p \log(k)$ (where k is the order of the problem; by assumption same for all students), then we get the same predictions. So there is no way to distinguish between the absolute values of student learning and intrinsic difficulty of problems.

On the other hand, the ordering of problems does not impact the estimation of relative learning rates (i.e., comparing students' learning rates, as reported in Fig. 6). To reliably detect the relative learning rates we just need them to be sufficiently different.

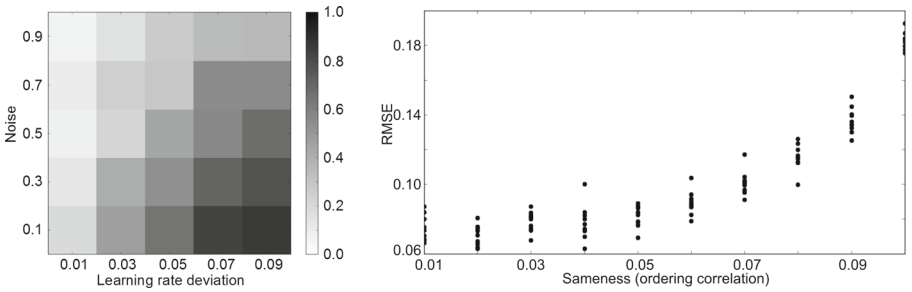

Fig. 6 *Left*: The color visualizes the Spearman's correlation coefficient between the true and computed values of the learning parameter δ. The experiment was run for varying noise and deviation of the learning parameter (with other meta-parameters fixed). *Right*: Estimation of absolute values of the learning rate. The x axis shows how much are the problem solving orders of individual students correlated, the y axis shows precision of estimation (measured by RMSE)

But how to deal with estimation of absolute values? We need to avoid bias from joint improvement of students, i.e., to have sufficiently diverse data. To measure "sameness" of problem solving order we use a mean correlation between students' ordering and mean ordering of all students. The higher this index the more coherent data are, thus the estimation of absolute values will be inaccurate. Figure 6 shows the dependence between this index and quality of absolute predictions (measured by RMSE).

Both information (relative and absolute predictions) are useful. Absolute predictions of learning rates can improve recommendations of problems and adapt pace of a tutor to a particular student. Absolute predictions can also serve as an evaluation of different problem sets designed for the same learning goal. Problem set with higher absolute learning rates should be preferred since it enables to make faster learning progression. Relative predictions are also useful since they can serve as a tool for teachers to determine learning trends in a class.

Evaluation of Predictions

Now we turn to experiments over real data from the Problem Solving Tutor. We compare model predictions with two simpler ways to predict problem solving times. At first, we consider the mean time[2] as a predictor – the simplest reasonable way to predict solving times. At second, we consider a simple 'personalized' predictor $\hat{t}_{sp} = m_p - \delta_s$, where m_p is the mean time for a problem p and δ_s is a "mean performance of student s with respect to other solvers", i.e., $\delta_s = (\sum m_p - t_{sp})/n_s$, where n_s is the number of problems solved by the student. Note that this corresponds to the initialization of parameter estimation of our basic model (Section "Stochastic Gradient Descent"); we call it a baseline model.

[2]Note that, consistently with the rest of the work, we compute the mean over the logarithm of time and thus the influence of outliers is limited and the mean is nearly the same as the median. For untransformed times the (quite common) use of mean time is inappropriate due to the skewness of the data.

Table 3 Quality of predictions for different models and problems measured by root mean square error metric

	Tilt	Rob.	Bin.	Reg.	Slith.	Sok.	Rush.	Nurik.
Mean time	1.037	1.382	1.249	1.366	1.194	1.275	1.072	1.161
Baseline	0.914	1.343	1.18	1.276	0.985	1.007	0.989	1.065
Basic model	0.911	1.321	1.158	1.276	0.959	0.998	0.977	1.062
Model w. variance	0.909	1.325	1.177	1.275	0.956	0.998	0.976	1.062
Model w. learning	0.941	1.339	1.194	1.312	0.979	1.016	0.991	1.081

Evaluation of model predictions was done by repeated random subsample cross-validation (student stratified). We performed 10 repetitions. In each run we choose randomly 70 % of students for whom we include all their data into the training set. For the remaining students we include the first 80 % of attempts into the training set and the last 20 % of attempts into the test set. Table 3. compares the results using the root mean square error metric (RSME). We have also evaluated other metrics like the Pearson and Spearman correlation coefficients and mean absolute error; the relative results are very similar.

The results show that the models provide clear improvement over the use of a mean time as a predictor. Most of the improvement is, however, captured by the baseline model. The additional complexity of the basic model brings a consistent improvement (the difference is statistically significant in all but two cases – Region puzzle and Nurikabe), but the effect size is much smaller than the difference between mean time and baseline model. Different variants (basic model with constant variance, individual variance, learning) of the model lead to similar predictions. The model with individual variance leads in same cases to improved RMSE, but in most cases the difference is not statistically significant. The model with learning even leads to slightly worse results than the basic model – on the current dataset the model with more parameters slightly overfits the data. By using a more fine tuned version of gradient descent (using different step sizes for individual parameters, particularly smaller step size for the parameter δ), the model with learning leads to improved predictions for some problems (Klusáček 2014), particularly the Slitherlink, which is a puzzle with many opportunities for improving performance.

Parameter Values for Real Data

Even though the more complex models do not lead to substantially improved predictions, they can still bear interesting information. Predictions are useful for guiding behaviour of the tutoring systems, but small improvement in prediction precision will not change the behaviour of the system in significant way. The important aim of the more complex models is to give us additional information about students and problems, e.g., the students' learning rate, which can be used for guiding the behaviour of tutoring system and for providing feedback to users. So we now analyse the estimates of parameter values and discuss their stability and usefulness.

For the parameter θ the results show that the estimated values are, as expected, approximately normally distributed. The variance of the distribution depends on the problem type – for educational problems we have larger variance of skills than for logic puzzles. There is a negative correlation between θ and the estimated σ (deviation of skill for individual attempts), i.e., students with lower skill have larger variability of their performance. This correlation differs for individual problem types, but typically it is in range from $r = -0.2$ to $r = -0.4$.

We have also studied the correlation between the problem parameters a, b, c. There is a slight correlation between the basic problem difficulty and its discrimination – more difficult problems are more discriminating ($r = 0.17$). Even weaker correlations are among the problem variance and difficulty ($r = 0.09$), resp. discrimination ($r = 0.16$).

Although there are some correlations among the parameters, generally the parameters are rather independent, i.e., each of them provides a useful information about the problem difficulty. For example, in intelligent tutoring system, it may be suitable to filter out problems with large variance or low discrimination.

Note that these results indirectly support the application of logarithmic transformation of times. If we had used untransformed times or some different transformation, there would be much stronger dependence.

Since the extended models do not improve predictions, it may be, however, that the additional parameters overfit the data and thus do not contain any valuable information. To test this hypothesis we performed the following experiment: we split the data into two disjoint halves, we use each half to train one model, and then we compare the parameter values in these two independent models. Specifically, we measure the Spearman correlation coefficient for values of each parameter.

Table 4 shows results for the model with learning. The results show, that estimates of basic difficulty and basic skill correlate highly, the weakest correlation between the estimates from the two halves is for the discrimination parameter. For students' learning rate, the additional parameter of the extended model, we get the correlation coefficient between 0.5 and 0.7 – a significant correlation which signals that the fitted parameters contain meaningful values.

Table 4 Spearman's correlation coefficient for parameter values obtained from two independent halves of the data

	Tilt	Rob.	Bin.	Reg.	Slith.	Sok.	Rush.	Nurik.
Skill θ	0.748	0.641	0.822	0.472	0.816	0.789	0.737	0.904
Learning rate δ	0.525	0.394	0.623	0.576	0.455	0.394	0.509	0.570
Difficulty b	0.994	0.961	0.951	0.927	0.981	0.963	0.962	0.837
Discrimination a	0.469	0.564	0.569	0.282	0.533	0.347	0.434	0.195

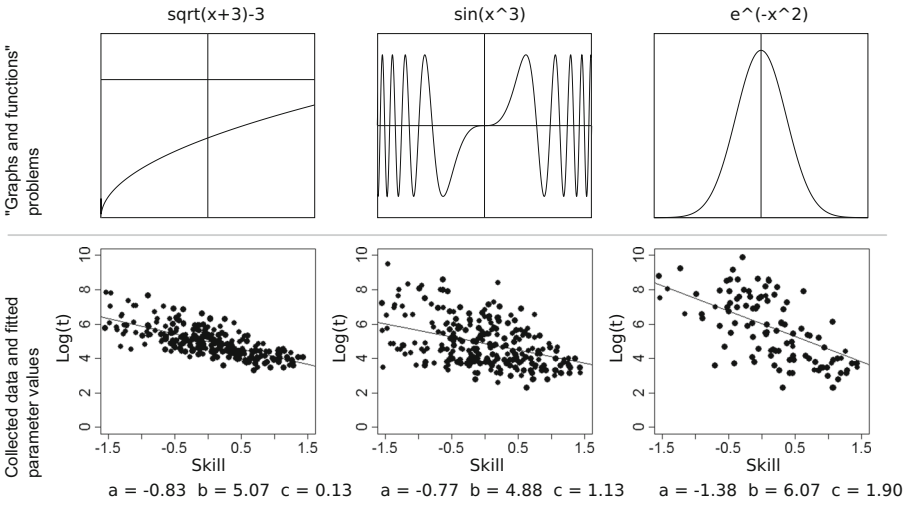

Fig. 7 "Graphs and functions" problem – three specific examples, for each of them we provide the collected data and values of parameters of the three parameter model

Insight Gained from Parameter Values

Let us illustrate the insight gained from the values of the model on a problem with graphs and functions – students are given a graph of a function and their task is to find a formula for the function. Figure 7 shows the collected data and values of model parameters for the three examples. The first and the second problem have similar difficulty, but the second one has much larger variance. The third problem is more difficult than the first two and is also more discriminating.

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

Detection of Concepts

So far we have assumed a single, independent latent problem solving skill for each problem type. This is, of course, a simplifying assumption. On one hand, for similar problems (e.g., Rush Hour and Sokoban puzzles) the problem solving performance is clearly related, and our results show that the estimated problem solving skills are

indeed highly correlated. For example for two of the mathematical problems the correlation between skills is 0.78, typical correlation between two problems is around 0.6. It may be advantageous to group together several similar problems and fit the data with one model with two dimensional skill.

On the other hand, in some cases it would be natural to assume multidimensional skill even for a single problem. For example in the case of the interactive graphs problem discussed above, some students may be proficient with polynomials, but struggle with trigonometric functions, and thus it may be useful to include at least 2 skills in the model.

To evaluate whether the model is able to detect different skills from the problem solving times, we performed the following experiment. We mix data from the Problem Solving Tutor for two different types of problems (e.g., two different logic puzzles) and remove information about the type of the problem from the data. Then we let an algorithm analyze the data and cluster problems into two groups – we fit the two-dimensional version of the model and use the learnt discrimination parameters to classify problems (see Section "Model with Multidimensional Skill"). The performance is measured by the number of correctly clustered problems (reported as percentage of all problems).

Table 5. presents the results. Note that the performance differs for individual problems. Results are very good (over 90 %, up to 97 %) for a Slitherlink puzzle, which is a puzzle where results strongly depend on logical reasoning skills and thus the noise in the data is low. On the other hand, results are poor for the Region division puzzle, where the noise in data is quite high since this puzzle is based more on insight and luck than on skill.

We have also experimented with an alternative technique (spectral clustering) for automatic detection of concepts (Boroš et al. 2013). The results were very similar to the results using the model of problem solving times. This suggest that the feasibility of automatic detection of concepts depends more on the quality of data

Table 5 Separating two problem types using the multidimensional model

	Sok.	Sli.	Nur.	Bin.	Til.	Rob.	Rus.	Reg.
Sokoban	–	97.1	89.3	94.6	89.4	80.5	82.9	71.7
Slitherlink	97.1	–	85.4	96.4	97.5	92.9	89.6	83.7
Nurikabe	89.3	85.4	–	87.7	90.1	89.4	84.9	76.5
Binary Crossword	94.6	96.4	87.7	–	96.0	82.1	91.2	81.1
Tilt Maze	89.4	97.5	90.1	96.0	–	81.6	91.3	76.0
Robotanist	80.5	92.9	89.4	82.1	81.6	–	79.8	67.5
Rushhour	82.9	89.6	84.9	91.2	91.3	79.8	–	71.3
Region Division	71.7	83.7	76.5	81.1	76.0	67.5	71.3	–
mean	*86.5*	*91.8*	*86.2*	*89.9*	*88.8*	*82.0*	*84.4*	*75.4*

The portion of problems that were classified correctly is given (in percent). The values shown are averages from multiple runs of the experiment. The mean accuracy over all problem pairs is 85.6 %

(size, noise) than on the specific choice of an algorithm. Boroš et al. (2013) also report on a specific example (a binary crossword puzzle) where the automatic detection of concepts bring interesting insight into the classification of problem instances (improving the expert classification included in the tutoring system). In an ongoing research we try to automatically combine expert opinion with student data (Nižnan et al. 2014).

Conclusions

We study problem solving in context of intelligent tutoring systems, particularly with the focus on timing information as opposed to just correctness of answers. This focus leads to different types of problems and requires new student models.

We propose models of students problem solving times, which assume a linear relationship between a problem solving skill and a logarithm of time. We describe two parameter estimation procedures. The model is related to two different areas: the item response theory and collaborative filtering. Analogically to models in item response theory, the described models have a group invariance property.

The model is already applied in an online Problem Solving Tutor to recommend problems of suitable difficulty. This system is already widely used (more than 400 000 problems solved), the collected data were used for evaluation of the model. The results show that the model brings only slight improvement compared to the baseline predictor, but also that the model provides interesting information about students and problems, e.g., discrimination of problems or learning rate of students. The model parameters may be useful for automatic problem selection and recommendation in intelligent tutoring systems and for providing feedback to students, teachers, or authors of educational materials. Extension of the model with multidimensional skill can be used for successful classification of problems based only on problem solving times.

Results of our experiments (e.g., residual analysis, analysis of parameter stability) support the appropriateness of the model for applications within tutoring systems (i.e., guiding the adaptive behaviour of the system, providing feedback to students, teachers, and system developers). The presented analysis does not address the issue of whether the detected student skill is a good estimate of problem solving abilities for external use (outside of the tutoring system), e.g., in high-stakes testing. In such context, a potential bias mentioned in Section "Normalization and Group Invariance" may become important. An interesting direction for further research would be to analyze the relation between the student skill as detected by the proposed model with other measures of problem solving ability, e.g., standarized tests or models using problems which combine both correctness of answers and timing information.

Acknowledgments We thank anonymous reviewers, who provided many comments and specific suggestions, which significantly improved the paper, and our collaborators, who have participated in some of the reported research: Petr Boroš, Juraj Nižnan, Matěj Klusáček, and Jiří Řihák. We also thank developers of the Problem Solving Tutor system, particularly Vít Stanislav.

References

Anderson, J., Boyle, C., & Reiser, B. (1985). Intelligent tutoring systems. *Science*, *228*(4698), 456–462.

Arroyo, I., Meheranian, H., & Woolf, B. P. (2010). Effort-based tutoring: An empirical approach to intelligent tutoring. In Proc. *of educational data mining* (Vol. 2010, pp. 1–10).

Baker, F. (2001). The basics of item response theory. University of Wisconsin.

Barnes, T. (2005). The q-matrix method: Mining student response data for knowledge. In *American association for artificial intelligence 2005 educational data mining workshop*.

Barnes, T., Bitzer, D., & Vouk, M. (2005). Experimental analysis of the q-matrix method in knowledge discovery. *Foundations of Intelligent Systems*, 11–41.

Beck, J., & Woolf, B. (2000). High-level student modeling with machine learning. In *Intelligent tutoring systems* (pp 584–593). Springer.

Beck, J.E. (2004). Using response times to model student disengagement. In *Proc. of the ITS2004 workshop on social and emotional intelligence in learning environments* (pp. 13–20).

Beck, J.E., & Chang, K.m. (2007). Identifiability: a fundamental problem of student modeling. In *User modeling* (pp. 137–146). Springer.

Beck, J.E., & Mostow, J. (2008). How who should practice: using learning decomposition to evaluate the efficacy of different types of practice for different types of students. In Intelligent tutoring systems (pp. 353–362). Springer.

Bergner, Y., Droschler, S., Kortemeyer, G., Rayyan, S., Seaton, D., & Pritchard, D. (2012). Model-based collaborative filtering analysis of student response data: machine-learning item response theory. In *Educational data mining* (pp. 95–102).

Bishop, C. (2006). *Pattern recognition and machine learning*. Springer.

Boroš, P., Nižnan, J., Pelánek, R., & Řihák, J. (2013). Automatic detection of concepts from problem solving times. In *Proc. of international conference on artificial intelligence in education* (Vol. 7926, pp. 595–598). Springer, LNCS.

Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: a survey. *ACM Computing Surveys (CSUR)*, *41*(3), 15.

Conati, C., Gertner, A., & Vanlehn, K. (2002). Using bayesian networks to manage uncertainty in student modeling. *User modeling and user-adapted interaction*, *12*(4), 371–417.

Csikszentmihalyi, M. (1991). *Flow: the psychology of optimal experience*. HarperPerennial New York.

De Ayala, R. (2008). *The theory and practice of item response theory*. The Guilford Press.

De La Torre, J. (2008). An empirically based method of q-matrix validation for the dina model: development and applications. *Journal of educational measurement*, *45*(4), 343–362.

Desmarais, M., Beheshti, B., & Naceur, R. (2012). Item to skills mapping: deriving a conjunctive q-matrix from data. In *Intelligent tutoring systems* (pp. 454–463). Springer.

Desmarais, M. C., & de Baker, R.SJ. (2012). A review of recent advances in learner and skill modeling in intelligent learning environments. *User Model User-Adapt Interact*, *22*(1–2), 9–38.

Fan, Z., Wang, C., Chang, H. H., & Douglas, J. (2012). Utilizing response time distributions for item selection in cat. *Journal of Educational and Behavioral Statistics*, *37*(5), 655–670.

Hirsch, W. Z. (1952). Manufacturing progress functions. *The Review of Economics and Statistics*, 143–155.

Jarušek, P. (2013). Modeling problem solving times in tutoring systems. PhD thesis, Faculty of informatics, Masaryk University Brno.

Jarušek, P., & Pelánek, R. (2011a). Problem response theory and its application for tutoring. In *Proc. of educational data mining* (pp. 374–375).

Jarušek, P., & Pelánek, R. (2011b). What determines difficulty of transport puzzles? In *Proc. of Florida artificial intelligence research society conference* (pp. 428–433). AAAI Press.

Jarušek, P., & Pelánek, R. (2012a). Analysis of a simple model of problem solving times. In *Proc. of intelligent tutoring systems* (Vol. 7315, pp. 379–388). Springer, LNCS.

Jarušek, P., & Pelánek, R. (2012b). Modeling and predicting students problem solving times. In *Proc. of international conference on current trends in theory and practice of computer science* (Vol. 7147, pp. 637–648). Springer, LNCS.

Jarušek, P., Klusáček, M., & Pelánek, R. (2013). Modeling students' learning and variability of performance in problem solving. In Proc. of international conference on educational data mining, international educational data mining society (pp. 256–259).

Kantor, P., Ricci, F., Rokach, L., & Shapira, B. (2010). *Recommender systems handbook*. Springer.

Klinkenberg, S., Straatemeier, M., & Van der Maas, H. (2011). Computer adaptive practice of maths ability using a new item response model for on the fly ability and difficulty estimation. *Computers & Education*, *57*(2), 1813–1824.

Klusáček, M. (2014). Modeling learning in problem solving, Master's thesis, Masaryk University.

Koedinger, K., Anderson, J., Hadley, W., & Mark, M. (1997). Intelligent tutoring goes to school in the big city. *International Journal of Artificial Intelligence in Education*, *8*(1), 30–43.

Koedinger, K., Corbett, A., Ritter, S., & Shapiro, L. (2000). Carnegie learning's cognitive tutor: summary research results. White paper Available from Carnegie Learning Inc 1200.

Koren, Y., & Bell, R. (2011). Advances in collaborative filtering. *Recommender systems handbook* (pp. 145–186).

Kotovsky, K., & Simon, H. (1990). What makes some problems really hard: explorations in the problem space of difficulty. *Cognitive Psychology*, *22*(2), 143–83.

Kotovsky, K., Hayes, J., & Simon, H. (1985). Why are some problems hard? Evidence from tower of Hanoi. *Cognitive Psychology*, *17*(2), 248–294.

LaBerge, D. (1975). Acquisition of automatic processing in perceptual and associative learning. *Attention and Performance*, *5*, 50–64.

Van der Linden, W. (2006). A lognormal model for response times on test items. *Journal of Educational and Behavioral Statistics*, *31*(2), 181.

Martin, B., Mitrovic, A., Koedinger, K. R., & Mathan, S. (2011). Evaluating and improving adaptive educational systems with learning curves. *User Modeling and User-Adapted Interaction*, *21*(3), 249–283.

Michlík, P., & Bieliková, M. (2010). Exercises recommending for limited time learning. *Procedia Computer Science*, *1*(2), 2821–2828.

Mostow, J., & Aist, G. (2001). Evaluating tutors that listen: an overview of project listen. In *Smart machines in education* (pp. 169–234). MIT Press.

Newell, A., & Rosenbloom, P. (1981). Mechanisms of skill acquisition and the law of practice. *Cognitive skills and their acquisition* (pp. 1–55).

Nižnan, J., Pelánek, R., & Řihák, J. (2014). Using problem solving times and expert opinion to detect skills. In *Educational data mining (EDM)* (pp. 21–27).

Pavlik, P., Bolster, T., Wu, S.M., Koedinger, K., & Macwhinney, B. (2008). Using optimally selected drill practice to train basic facts. In *Intelligent tutoring systems* (pp. 593–602). Springer.

Pavlik, P. I., & Anderson, J. R. (2005). Practice and forgetting effects on vocabulary memory: an activation-based model of the spacing effect. *Cognitive Science*, *29*(4), 559–586.

Pavlik, P. I., & Anderson, J. R. (2008). Using a model to compute the optimal schedule of practice. *Journal of Experimental Psychology: Applied*, *14*(2), 101.

Pelánek, R. (2011). Difficulty rating of sudoku puzzles by a computational model. In *Proc. of Florida artificial intelligence research society conference*.

Pizlo, Z., & Li, Z. (2005). Solving combinatorial problems: the 15-puzzle. *Memory and Cognition*, *33*(6), 1069.

Rai, D., & Beck, J. (2011). Exploring user data from a game-like math tutor: a case study in causal modeling. In Pechenizkiy, M., et al. (Eds.), *Educational data mining* (pp. 307–313).

Rupp, A., & Templin, J. (2008). The effects of q-matrix misspecification on parameter estimates and classification accuracy in the dina model. *Educational and Psychological Measurement*, *68*(1), 78–96.

Simon, H., & Newell, A. (1972). *Human problem solving*. Prentice Hall.

Šormová, H. (2014). Detekce anomálií v datech o řešení problému (Anomaly detection for problem solving data). Master's thesis, Masaryk University.

Tatsuoka, K. (1983). Rule space: an approach for dealing with misconceptions based on item response theory. *Journal of Educational Measurement*, *20*(4), 345–354.

Thai-Nghe, N., Drumond, L., Horváth, T., Krohn-Grimberghe, A., Nanopoulos, A., & Schmidt-Thieme, L. (2011). Factorization techniques for predicting student performance. Educational recommender systems and technologies: practices and challenges IGI Global.

Van Der Linden, W. (2009). Conceptual issues in response-time modeling. *Journal of Educational Measurement*, *46*(3), 247–272.

Vanlehn, K. (2006). The behavior of tutoring systems. *International Journal of Artificial Intelligence in Education*, *16*(3), 227–265.

Wang, Y., & Heffernan, N. (2012). Leveraging first response time into the knowledge tracing model. In *Proc. of international conference on educational data mining* (pp. 176–179).