Bayesian networks for the test score prediction: a case study on a math graduation exam

Martin Plajner[1] and Jiří Vomlel[1,2]

[1] Institute of Information Theory and Automation,
Czech Academy of Sciences,
Pod Vodárenskou věží 4, Prague 8, Czechia

[2] Faculty of Management,
University of Economics, Prague
Jarošovská 1117/II, 377 01 Jindřichův Hradec, Czechia

vomlel@utia.cas.cz

Abstract. In this paper we study the problem of student knowledge level estimation. We use probabilistic models learned from collected data to model the tested students. We propose and compare experimentally several different Bayesian network models for the score prediction of student's knowledge. The proposed scoring algorithm provides not only the expected value of the total score but the whole probability distribution of the total score. This means that confidence intervals of predicted total score can be provided along the expected value. The key that enabled efficient computations with the studied models is a newly proposed inference algorithm based on the CP tensor decomposition, which is used for the computation of the score distribution. The proposed algorithm is two orders of magnitude faster than a state of the art method. We report results of experimental comparisons on a large dataset from the Czech National Graduation Exam in Mathematics. In this evaluation the best performing model is an IRT model with one continuous normally distributed skill variable related to all items by the graded response models. The second best is a multidimensional IRT model with an expert structure of items-skills relations and a covariance matrix for the skills. This model has a higher improvement with larger training sets and seems to be the model of choice if a sufficiently large training dataset is available.

Keywords: Bayesian networks · Educational Testing · Score Prediction · Efficient Probabilistic Inference · Multidimensional IRT · CP tensor decomposition.

1 Introduction

In this paper we study the problem of student knowledge level estimation. For this problem we use probabilistic models learned from collected data. When a model is applied to testing a particular student the model is updated by items answered by the tested student so far and the updated model is used to select the

next item to be answered by the tested student. This concept is often referred as computerized adaptive testing (CAT), see, e.g. [11, 20].

In their seminal work [2] Almond and Mislevy pointed out that the students can be modelled well using Bayesian networks [14, 9, 10]. This idea was further elaborated in [19, 1]. All probabilistic models studied in this paper can be considered Bayesian networks (BNs) in a broader sense.

The paper is organized as follows. In Section 2 we describe probabilistic models that can be used to model student knowledge. In Section 3 we explain how the probabilistic inference with these models can be performed. Section 4 represents the main theoretical contribution of the paper, which is an efficient method for the score computation based on the CP tensor decomposition. In Section 5, which presents the main experimental contribution of the paper, we report results of experimental comparisons using a large dataset from the Czech National Graduation Exam in Mathematics. The results are summarized in Section 6.

2 Models

We will model the problem using models sharing four groups of variables:

- *Skills* (in some models called Factors), denoted $S_1, \ldots, S_m$ are unobserved (hidden) variables used to specify student abilities (skills). They will be either binary variables, discrete ordinal variables, or continuous variables depending on the model used.
- *Items* (also called Questions or Tasks), denoted $X_1, \ldots, X_n$ will be discrete ordinal variables whose states correspond to the number of points received for the answer. We will assume the states (points) are from a set of integers $\{0, 1, \ldots, z-1\}$. When testing a particular student we will estimate their probability distributions using the collected evidence.
- *Answered items* are a subset of copies of items $X_1, \ldots, X_n$. An answered item is observed for the currently tested student. Answered item corresponding to an item $X_i, i \in \{1, \ldots, n\}$ is denoted X'_i. Conditional probability of each X'_i given the skills is by definition the same as for their corresponding item X_i. The answered items are used to propagate collected evidence to the model[3].
- *The total score node*, denoted Y, which represents the total sum of scores from all questions[4]. The values of Y are all possible total sums of items' values of $X_1, \ldots, X_n$. For example, if all items are binary (taking values 0 and 1) then Y has $n+1$ values $(0, 1, \ldots, n)$.

[3] For each answered item X'_i its corresponding item X_i is generally still uncertain. Items X_i represent certain type of a question or a task that can be repeated and the result need not to be the same for the same student, i.e., we estimate student's skills and admit mistakes even if he/she has required skills and vice versa.

[4] The score we estimate is not the score the tested student gets at the end of the current test but in another test of the same type. Of course, the model can also provide the estimate of the test score in the current test – in this case, another score node, say Y', would have nodes X'_i as its parents instead. In this case all methods would necessarily have the zero score prediction error at the end of the test.

The general structure of all models will be similar albeit they will differ by the number of skill variables, by the type of skill variables, and by a possible direct dependence among skills represented in the model. In Figure 1 and Figure 2 we give examples of two types of considered models.

The BN model in Figure 1 has three independent skills. These skills influence scores of items. It is assumed that for each item a domain expert decided which skills are relevant. In the model of Figure 1 skill S_1 influences item X_1, skill S_2 influences both X_2 and X_3 items, and skill S_3 also influences both items X_2 and X_3 but also item X_4. The items X_1, X_2, X_3, and X_4 are never observed while items X_1', X_2', X_3', and X_4' represent observed items and are included in the model only if they are observed. If they are not observed then they are omitted from the model since they represent barren variables.

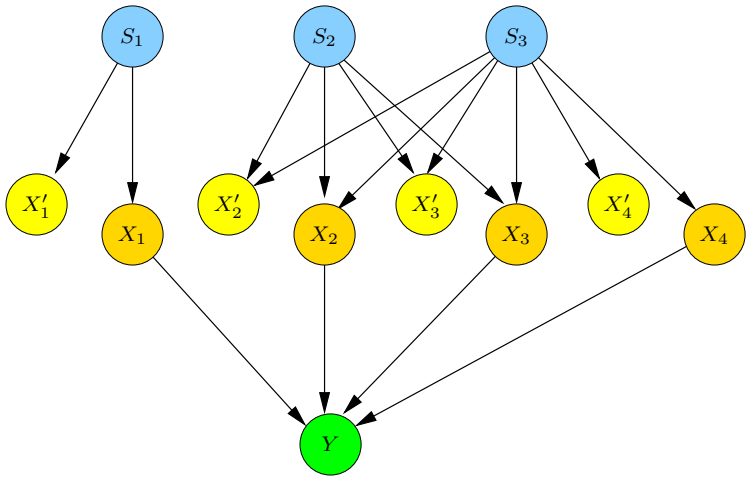

Fig. 1. An expert BN model with independent skills

Another BN model is presented in Figure 2. This differs from the BN model in Figure 1 in two aspects. First, in this models all items are related to all skills. Second, the skills are dependent. Typically, this model would be learned from collected data without using expert knowledge.

The models discussed above can be combined. For example, expert knowledge of relations of items to skills from the BN model of Figure 1 can be used to reduce the number of edges in the BN model of Figure 2.

In this paper we will compare following models:

– An expert model with independent skills (Figure 1 type). Skills are represented by ordinal variables with three states. Items are related to their parent skills by general conditional probability tables restricted by monotonicity conditions [5]. The model parameters will be learned using restricted gradient method [15]. This model will be referred as *rgrad*. We should note that other

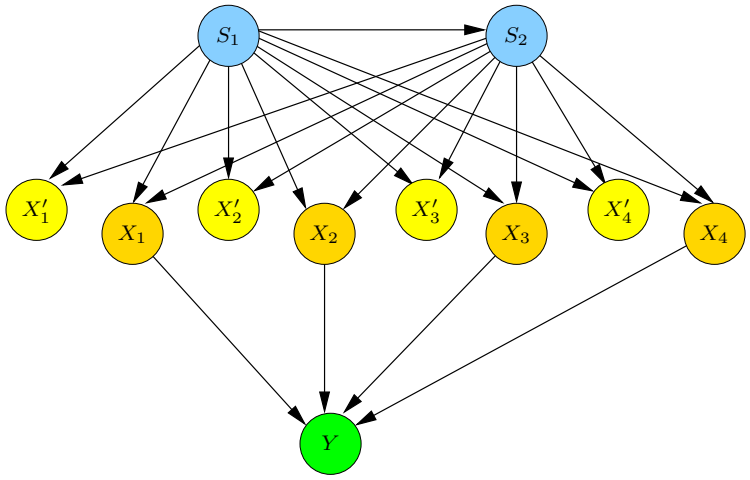

Fig. 2. A BN model with dependent skills related to all items

methods for learning conditional probability tables satisfying monotonicity conditions could be also used, e.g. the isotonic regression EM by Masegosa et al. [12]. We decided to report results of the restricted gradient method in this paper since it provided best results in preliminary experiments [16].

- A model with one continuous skill variable related to all items. The skill variable is assumed to have Gaussian distributions and items are related to the skill node by the graded response models (GRMs) [17]. This model belongs to the well-known family of IRT models and, thus, it will be referred as *irt*.
- An expert model with the same structure as *rgrad* but the skill variables are continous, each having the Gaussian distributions and the items are related to skills by GRMs as in IRT. This model can be considered as an example of a multidimensional IRT [8] with the zero covariance among skills (often called factors in IRT). Therefore it will be referred as *mirt-cov0*.
- An expert model with the same structure as *rgrad* with continous skill variables which differs from *mirt-cov0* only by including relation between skills represented by multidimensional Gaussian distribution with a covariance matrix allowing nonzero non-diagonal elements. This model will be referred as *mirt-cov1*.
- The last group of models has the structure of model from Figure 2, i.e., all items are related to all skills and the skills are dependent. It is a straightforward generalization of *irt* to more skills (factors). Skills are assumed to have multidimensional Gaussian distribution with the covariance matrix allowing nonzero non-diagonal elements. Depending on the number of factors these models will be referred as *mirt-2F*, *mirt-3F*, and *mirt-4F*, respectively.

The model parameters of the IRT and the multidimensional IRT models will be learned using algorithms implemented in the R mirt package [4].

The joint probability distribution of these models is defined by their structure represented by directed acyclic graph and their conditional probability distributions of each node given its parents as:

$$P(Y|X_1,\ldots,X_n)\cdot\prod_{i=1}^{n}P(X_i|pa(X_i))\cdot\prod_{i=1}^{n}P(X_i'|pa(X_i'))\cdot\prod_{j=1}^{m}P(S_j|pa(S_j))\ .$$

3 Probabilistic Inference

Models introduced in Section 2 will be used to estimate the probability distribution of the total score given the items answered so far. We will use symbol I to denote the set of indexes of already answered items X_i'. To simplify notation we will use symbol e to denote the evidence collected so far, i.e., $e=\{X_i'=x_i'\}_{i\in I}$.

The probability distribution of the total score given evidence e is (in case of discrete skill nodes) computed as:

$$P(Y|e)=\sum_{X_1,\ldots,X_n}\sum_{S_1,\ldots,S_m}P(Y,X_1,\ldots,X_n,S_1,\ldots,S_m|e)\ .$$

For the continuous skill nodes the integrals are used instead of the summation.

Using the general structure of the model this can be rewritten as:

$$P(Y|e)=\sum_{X_1,\ldots,X_n}\sum_{S_1,\ldots,S_m}P(Y|X_1,\ldots,X_n)\cdot Q(X_1,\ldots,X_n,S_1,\ldots,S_m|e)$$

where

$$Q(X_1,\ldots,X_n,S_1,\ldots,S_m|e)=\prod_{i=1}^{n}P(X_i|pa(X_i))\cdot R(S_1,\ldots,S_m|e)$$

$$R(S_1,\ldots,S_m|e)\propto\prod_{i\in I}P(X_i'=x_i'|pa(X_i'))\cdot\prod_{j=1}^{m}P(S_j|pa(S_j))$$

and the conditional probability distribution

$$P(Y=y|X_1=x_1,\ldots,X_n=x_n)=\begin{cases}1 & \text{if } y=\sum_{i=1}^{n}x_i\\ 0 & \text{otherwise}\end{cases}$$

is deterministic and represents the distribution of the total sum of items' values.

In all considered models the values of items are dependent through the skills. Therefore we cannot sum out the skills for each item independently and then compute the probability distribution of the total sum. On the other hand, if a skill configuration is fixed, say it is a vector $s=(s_1,\ldots,s_m)$, then items become independent. This means that for each configuration of skills we can write

$$Q(X_1,\ldots,X_n,S_1=s_1,\ldots,S_m=s_m|e)=\prod_{i=1}^{n}P(X_i|s)\cdot R(s|e)\ .$$

Using the above formula and denoting by x the vector $(x_1, \ldots, x_n)$, by $\mathcal{X} = \times_{i=1}^{n} \mathcal{X}_i$ the set of all configurations of x, and by $\mathcal{S}$ the set of all configurations of all skills we get:

$$P(Y|e) = \sum_{s \in \mathcal{S}} \left(\sum_{x \in \mathcal{X}} P(Y|X_1 = x_1, \ldots, X_n = x_n) \cdot \prod_{i=1}^{n} P(X_i = x_i|s) \right) \cdot R(s|e) \ .$$

In case of continuous skill nodes we will approximate the integrals over skills by a sufficiently large finite set of skill configurations $s = (s_1, \ldots, s_m)$ chosen so that they cover well the space of skill values. For this purpose, we will use Halton sequences [6] in m dimensions to generate points in the Cartesian product $\mathcal{S}$ of skills' state spaces[5]. Although these sequences are deterministic, they are of low discrepancy, i.e. they cover well the space $\mathcal{S}$. Thus, we can write:

$$P(Y|e) = \sum_{s \in \mathcal{S}} P(Y|s) \cdot R(s|e) \ , \text{ where} \tag{1}$$

$$P(Y|s) = \sum_{x \in \mathcal{X}} P(Y|X_1 = x_1, \ldots, X_n = x_n) \cdot \prod_{i=1}^{n} P(X_i = x_i|s) \ . \tag{2}$$

Now, the only remaining obstacle for efficient inference is the conditional distribution $P(Y|X_1 = x_1, \ldots, X_n = x_n)$, which can be very large since, typically, the models contain tens to hundreds of items. An efficient transformation of this distribution will be the topic of the next section.

4 Efficient Inference Method for the Score Computation

In this section we present a computationally efficient method that will be used to compute the probability distribution of the total score. It is based on the CP tensor decomposition [7, 3] and can be also viewed as an application of Discrete Fourier Transformation (DFT). As we will show latter it is several orders of magnitude faster than a standard probabilistic inference approach based on, so called, parent divorcing [13]. The proposed method is especially useful for discrete random variables that have a large number of parents. This is the case of our case study since the total score variable has 37 discrete valued parents.

We assume each of n items takes values (points) from the set $\{0, 1, \ldots, z-1\}$. Therefore the maximum possible total score is $n(z-1)$ and the total number of states of Y is[6] $k = 1 + n(z-1)$. In the derivation of our algorithm we will exploit Theorem 3 and its proof given in [18]. From this result it follows that for all $y, x_1, \ldots, x_n$ we can write[7]:

$$P(Y = y|X_1 = x_1, \ldots, X_n = x_n)$$

[5] In order to use the same formula for both discrete and continuous skills we will use $\mathcal{S}$ to denote the set of skill configurations also in the continuous case.

[6] Please, note we include also the state 0.

[7] Please, note that the upper index m of $\alpha_j^m, j = 1, \ldots, k.$ represents the power of α_j.

$$= \sum_{b=1}^{k} \alpha_b^{x_1} \cdot \ldots \cdot \alpha_b^{x_n} \cdot \beta_{b,y} \quad = \quad \sum_{b=1}^{k} \alpha_b^{x_1+\ldots+x_n} \cdot \beta_{b,y} \ , \tag{3}$$

where $\alpha_1, \ldots, \alpha_k$ are pairwise distinct real or complex numbers and values of $\beta_{b,y}$ are solutions of the system of linear equations

$$Y = AX \ , \tag{4}$$

where Y is the $k \times k$ identity matrix, A is the $k \times k$ Vandermonde matrix defined by a vector $\alpha = (\alpha_1, \ldots, \alpha_k)$ as

$$A = \left\{\alpha_i^j\right\}_{i=1,j=0}^{k,k-1} \quad = \quad \begin{pmatrix} \alpha_1^0 & \alpha_2^0 & \ldots & \alpha_k^0 \\ \alpha_1^1 & \alpha_2^1 & \ldots & \alpha_k^1 \\ \ldots & & & \\ \alpha_1^{(k-1)} & \alpha_2^{(k-1)} & \ldots & \alpha_k^{(k-1)} \end{pmatrix} \ ,$$

and X is the matrix defined as

$$X = \{\beta_{b,y}\}_{b=1,y=0}^{k,k-1} \quad = \quad \begin{pmatrix} \beta_{1,0} & \beta_{1,1} & \ldots & \beta_{1,k-1} \\ \beta_{2,0} & \beta_{2,1} & \ldots & \beta_{1,k-1} \\ \ldots & & & \\ \beta_{k,0} & \beta_{k,1} & \ldots & \beta_{k,k-1} \end{pmatrix} \ .$$

Using the standard notation for probability tables we can write the equation (3) in a compact form as

$$P(Y|X_1, \ldots, X_n) = \sum_B P(Y, B) \cdot \prod_{i=1}^{n} P(X_i, B) \ , \tag{5}$$

where for all values b, x_i, y of variables B, X_i, Y, respectively, it holds:

$$P(Y = y, B = b) = \beta_{b,y} \quad \text{and} \quad P(X_i = x_i, B = b) = \alpha_b^{x_i} \ .$$

To achieve a good numerical stability for large dimensional problems (i.e., with high values of n) it is wise to use complex valued coefficients $\alpha_1, \ldots, \alpha_k$. We define them as the complex numbers from the unit circle (roots of unity) in the space of complex numbers, i.e., for $j = 1, \ldots, k$:

$$\alpha_j = \exp\left(j \cdot \frac{2\pi i}{k}\right) \ , \tag{6}$$

where i is the imaginary unit, satisfying the equation $i^2 = -1$. This brings also a very nice benefit since, in this case, the solution of the system of linear equations (4) is well-known:

$$X = \frac{1}{k} \cdot \overline{A^T} \ ,$$

where A^T denotes the transpose of A and $\overline{A^T}$ denotes the matrix with complex conjugated entries of A^T.

Remark 1. Vandermonde matrix A with entries specified by equation (6) is actually the Discrete Fourier Transformation (DFT) matrix.

Now we are ready to specify the final step of the probabilistic inference discussed in Section 3. In formula (2) we replace the conditional probability distribution of Y by the expression specified by formula (5)

$$\begin{aligned} P(Y|s) &= \sum_{x\in\mathcal{X}} \sum_{b\in\{1,\dots,k\}} P(Y, B=b) \cdot \prod_{i=1}^{n} P(X_i = x_i, B=b) \cdot \prod_{i=1}^{n} P(X_i = x_i|s) \\ &= \sum_{b\in\{1,\dots,k\}} P(Y, B=b) \cdot \prod_{i=1}^{n} \sum_{x_i\in\mathcal{X}_i} P(X_i = x_i, B=b) \cdot P(X_i = x_i|s) \\ &= \sum_{b\in\{1,\dots,k\}} \beta_{b,y} \cdot \prod_{i=1}^{n} \sum_{x_i\in\mathcal{X}_i} \alpha_b^{x_i} \cdot P(X_i = x_i|s) \ . \end{aligned}$$

```
score ← function(P)
{
  z ← nrow(P)
  n ← ncol(P)
  k ← 1+n*(z−1)
  alpha ← complex(modulus=1,argument=(1:k)*2*pi/k)
  A ← outer(alpha,seq(0,k−1),'^')
  X ← (1/k) * t(Conj(A))
  v ← rep(1,k)
  for (i in 1:n){
    v ← v * crossprod(t(A[,1:z]),P[,i])
  }
  y ← drop(Re(crossprod(t(X),v)))
  return(y)
}
```

Fig. 3. Scoring algorithm for a given skill configuration

In Figure 3 we present[8] the algorithm for the total score computation for a given skill configuration s. The algorithm input is specified in $z \times n$ matrix[9] P defined by distributions $P(X_i|s)$ so that the element of j^{th} row and i^{th} column

[8] We we use the notation of R, in which we implemented the algorithm and which we believe is self-explanatory. The functions **outer** and **crossprod** implement the outer and the matrix products, respectively. The third argument of **outer** gives the function to use on the outer products, in this case it is the exponentiation.

[9] The number of rows z is the maximum item value computed over all items plus one. If a value is impossible then the corresponding matrix entry is set to zero.

of matrix P corresponds to the value $P(X_i = j|s)$ of distribution $P(X_i|s)$. The output y of the algorithm is $P(Y|s)$ – the probability distribution of the total score for the given s.

Proposition 1. *The algorithm presented in Figure 3 computes the probability distribution of the total score $P(Y|s)$ in $O((nz)^2)$ time.*

Proof. The most computationally demanding steps of the algorithm are the matrix products **crossprod**. The first matrix product is repeated n times in the for loop of the algorithm and it is applied to a matrix with dimensions $k \times z$ and a vector of length z, which requires kz multiplications and additions. Since the cross product is repeated n times and $k = nz$ the overall complexity of this step is $O((nz)^2)$. The second matrix product is applied to a matrix with dimensions $k \times k$ and a vector of length k, which implies the overall complexity of this step is also $O((nz)^2)$. Remaining steps of the algorithm have a lower complexity therefore the computational complexity of the algorithm is $O((nz)^2)$.

To estimate the total score of a particular student the scoring algorithm given a skill configuration s is repeated for each configuration s of student's skills, which can be thousands or more. Therefore, the complexity of the scoring algorithm represents an important issue. The complexity of the presented algorithm is quadratic with respect to the number of possible total scores.

In Figure 4 we compare the computational CPU time of the presented scoring algorithm based on the CP tensor decomposition with the state of the art method – the parent divorcing method [13]. In the figure the horizontal axis corresponds to the number of items that are parents of the score node. The items corresponds to items from the model of the Czech National Graduation Exam in Mathematics described in Section 5. They are added one by one in the ascending order. Most items have two states only, but some of them have more than two states (the maximum is four). The vertical axis corresponds to the total CPU time of one thousand of actual computations of the score for the given number of items[10]. We can see that the scoring algorithm based on the CP tensor decomposition is two orders of magnitude faster than the parent divorcing method.

In Figure 5 the computational CPU time of the presented scoring algorithm based on the CP tensor decomposition. The vertical axis corresponds to the average CPU time of one computation of the score for the given number of items. This time the scale is linear to better reveal the quadratic computational complexity of the algorithm.

5 Experimental model comparisons

In this section we report results of experimental comparisons performed using a large dataset from the Czech National Graduation Exam in Mathematics. This dataset contains answers from more than 20,000 students who took this test

[10] Please, note the vertical axis has a logarithmic scale.

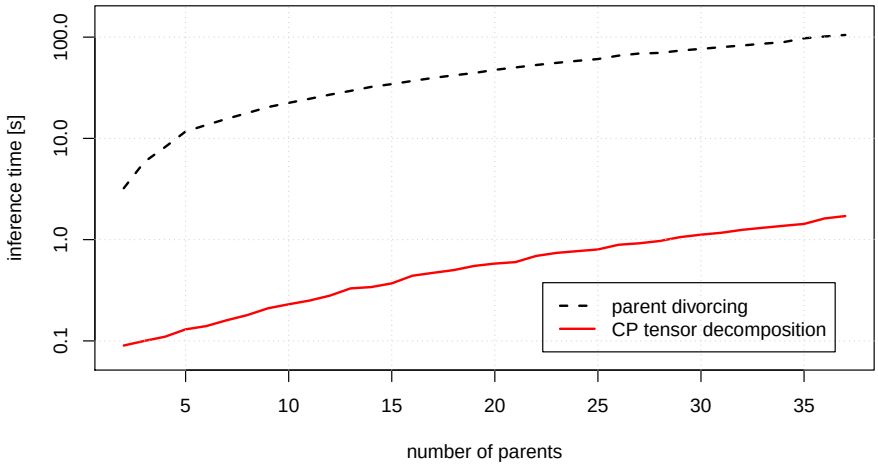

Fig. 4. Comparisons of the total computational time of 1000 computations of the scoring algorithms using the logarithmic time scale.

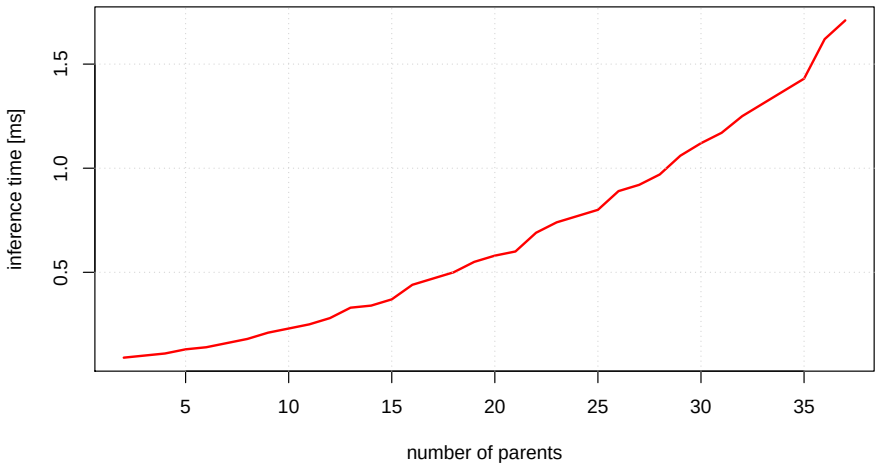

Fig. 5. The average computational time (in milliseconds) of the scoring algorithm based on the CP tensor decomposition using the linear time scale.

in the year 2015. We randomly selected training subsets of different sizes and a testing subset consisting of answers of 100 students[11]. The test contains 37 items, most of them scored with either 0 or 1 point, the highest number of points for one item is 3. The expert model contains 8 skill nodes and is described in [15].

Model comparisons are presented in Figure 6. We evaluate the score prediction quality during adaptive tests performed for each of 100 students from the testing set. The criteria used for evaluation was the average absolute difference of the expected score computed by each model and the true score. The items were selected for each student using an adaptive criteria. To get comparable results we used the same sequence of items for all models. The sequence was established using the *rgrad* model with the expected information gain criteria[12].

From the plots we can clearly see that the best performing algorithm is *irt* for both sizes of the training dataset[13]. The second best performing algorithm was *mirt-COV1*. This is the algorithm that uses the expert structure for items-skills relations and includes relations between skills. It is important to note that the improvement with larger training set is larger for *mirt-COV1* than for *irt*. Generally, more complex models seem to require more training data than *irt* to achieve better performance. Also, expert models that do not represent dependence between skills (*mirt-COV0* and *rgrad*) have worse performance than the corresponding expert model representing this dependence (*mirt-COV1*). This indicates that the current expert model *rgrad* could be probably improved by including such dependencies explicitly in the model. Finally, from models with more than one skill those that have the expert structure for items-skills relations perform better than those that have not.

Since the scoring algorithm provides not only the expected value of the total score but the whole probability distribution of the total score it can be used to provide confidence intervals of predicted total score as well. In Figure 7 we present an example of 95% confidence interval for one student from the testing set using the *irt* model learned from 160 students' records from training data.

6 Discussion

In this paper we presented an experimental comparison of different Bayesian network models for score prediction during an adaptive test of student's knowledge. The key that enabled efficient computation was a new inference algorithm used for computation of the score distribution. We are not aware of any other work that would compare several diverse student models as those discussed in this paper on a real data.

[11] Of course, the selection process ensured that the datasets were disjoint.

[12] Albeit the item sequences might slightly differ for different models we do not expect this has a significant impact on models' performance. The score prediction is independent of the order of items in the sequence if the items are the same.

[13] All claims about algorithms' performance were verified using the Wilcoxon rank test with significance level 0.01. Actually, the probability of the alternative hypothesis was always lower than 10^{-15}.

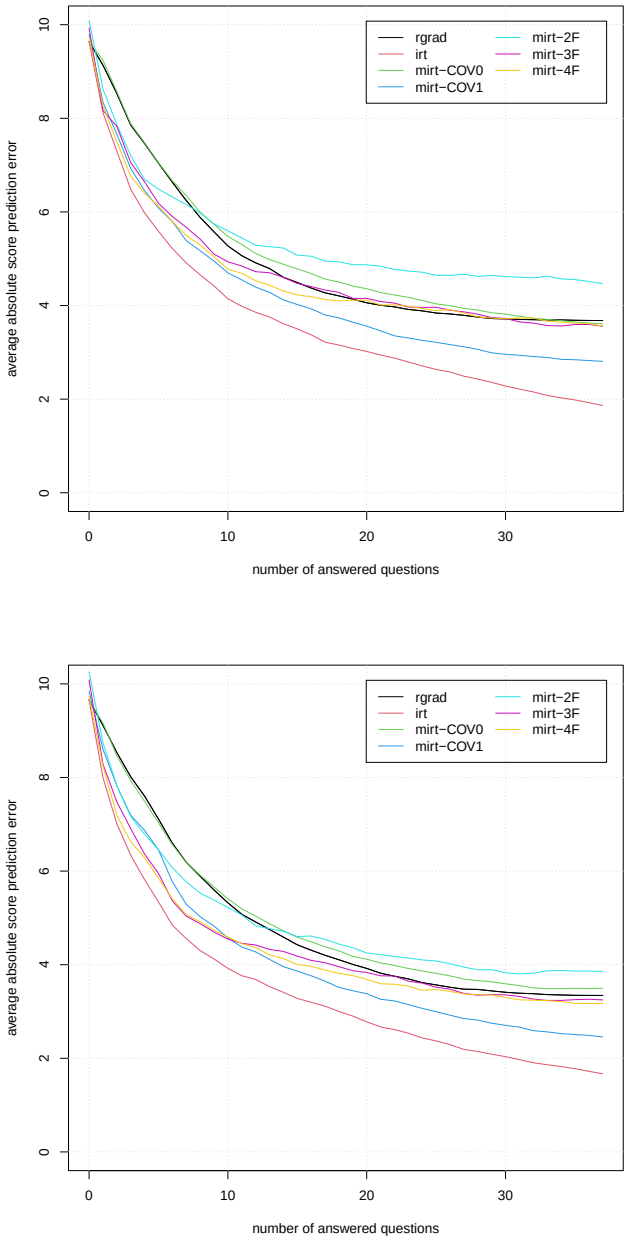

Fig. 6. Model comparisons for the training set consisting of 160 and 640 students, respectively.

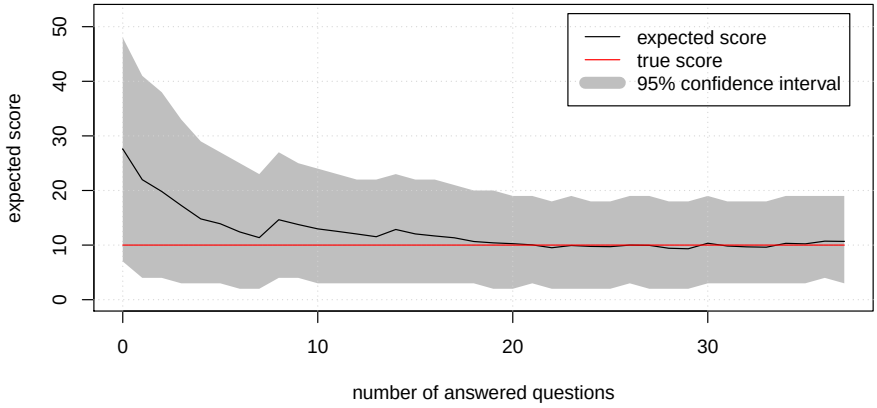

Fig. 7. Expected total score and 95% confidence interval for one student from the testing set.

The models presented in this paper were tested solely with respect to their ability to predict the total score. It should be noted that another important task in educational testing is skill analysis. For example, the estimation of the level of student's skills plays an important role in intelligent tutoring systems. Only some of the tested models contain skills with a comprehensible interpretation. In *rgrad*, *mirt-cov0*, and *mirt-cov1* experts created the model structures with such an interpretation in mind. On the other hand, skills in models *mirt-2F*, *mirt-3F*, and *mirt-4F* are truly hidden variables and do not have any clear interpretation. In the specific case of the *irt* model the skill node can be interpreted as a general ability to answer/solve the modeled items.

Since the data used in our study do not provide information about presence/absence of skills of a student we cannot test directly models' skill prediction quality. However, we can assume that skill and total score prediction quality are related. We expect that best score prediction models with expert-identified skills can be used also as good skill analysis models.

Acknowledgements

We would like to thank anonymous reviewers for their comments and suggestions that helped us to improve the paper. This work was supported by the Czech Science Foundation, Project No. 19-04579S.

References

1. Almond, R.G., Mislevy, R.J., Steinberg, L.S., Yan, D., Williamson, D.M.: Bayesian Networks in Educational Assessment. Springer New York (2015)
2. Almond, R.G., Mislevy, R.J.: Graphical models and computerized adaptive testing. Applied Psychological Measurement **23**(3), 223–237 (1999)
3. Carroll, J.D., Chang, J.J.: Analysis of individual differences in multidimensional scaling via an n-way generalization of Eckart-Young decomposition. Psychometrika **35**, 283–319 (1970)
4. Chalmers, R.P.: mirt: A multidimensional item response theory package for the R environment. Journal of Statistical Software, Articles **48**(6), 1–29 (2012), https://doi.org/10.18637/jss.v048.i06
5. van der Gaag, L.C., Bodlaender, H.L., Feelders, A.J.: Monotonicity in bayesian networks. In: Proceedings of the Twentieth Conference on Uncertainty in Artificial Intelligence. pp. 569–576. AUAI Press (2004)
6. Halton, J.H.: Algorithm 247: Radical-inverse quasi-random point sequence. Communication of the ACM **7**(12), 701–702 (1964), https://doi.org/10.1145/355588.365104
7. Harshman, R.A.: Foundations of the PARAFAC procedure: Models and conditions for an "explanatory" multi-mode factor analysis. UCLA Working Papers in Phonetics **16**, 1–84 (1970)
8. Hartig, J., Höhler, J.: Multidimensional IRT models for the assessment of competencies. Studies in Educational Evaluation **35**(2), 57–63 (2009), https://doi.org/10.1016/j.stueduc.2009.10.002
9. Jensen, F.V.: Bayesian networks and decision graphs. Springer-Verlag, New York (2001)
10. Lauritzen, S.L., Spiegelhalter, D.J.: Local computations with probabilities on graphical structures and their application to expert systems (with discussion). Journal of the Royal Statistical Society, Series B **50**, 157–224 (1988)
11. van der Linden, W.J., Glas, C.A.W. (eds.): Elements of Adaptive Testing. Springer NY (2010)
12. Masegosa, A.R., Feelders, A.J., van der Gaag, L.C.: Learning from incomplete data in Bayesian networks with qualitative influences. International Journal of Approximate Reasoning **69**, 18–34 (2016). https://doi.org/https://doi.org/10.1016/j.ijar.2015.11.004
13. Olesen, K.G., Kjaerulff, U., Jensen, F., Jensen, F.V., Falck, B., Andreassen, S., Andersen, S.K.: A Munin network for the median nerve – a case study on loops. Applied Artificial Intelligence **3**(2-3), 385–403 (1989), https://doi.org/10.1080/08839518908949933
14. Pearl, J.: Probabilistic reasoning in intelligent systems: networks of plausible inference. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1988)
15. Plajner, M., Vomlel, J.: Learning bipartite Bayesian networks under monotonicity restrictions. International Journal of General Systems **49**(1), 88–111 (2020), https://doi.org/10.1080/03081079.2019.1692004
16. Plajner, M., Vomlel, J.: Monotonicity in practice of adaptive testing (2020), https://arxiv.org/abs/2009.06981
17. Samejima, F.: Estimation of latent ability using a response pattern of graded scores. Psychometrika **34**, 1—97 (1969), https://doi.org/10.1007/BF03372160
18. Savicky, P., Vomlel, J.: Exploiting tensor rank-one decomposition in probabilistic inference. Kybernetika **43**(5), 747–764 (2007)

19. Vomlel, J.: Bayesian networks in educational testing. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems **12**(supp01), 83–100 (2004)
20. Wainer, H., Dorans, N.J.: Computerized Adaptive Testing: A Primer. Routledge (1990)