Two applications of Bayesian networks*

Jiří Vomlel[1,2]
vomlel@utia.cas.cz

[1] Laboratory for Intelligent Systems
University of Economics,
Prague, Czech Republic

[2] Institute of Information Theory and Automation
Academy of Sciences of the Czech Republic
Prague, Czech Republic

Abstract. We present two recent applications of Bayesian networks: adaptive testing and troubleshooting man-made devices. We review briefly the underlying theory and provide a general framework for building strategies using Bayesian network models. We discuss applications of the framework to adaptive testing and troubleshooting. The paper is based on our experience with two projects: a student semester project during which an adaptive test of students' knowledge of operations with fractions was designed and the SACSO project - a joint project of Hewlett-Packard and Aalborg University focused on development of methods for troubleshooting complex electro-mechanical systems.

1 Introduction

In this section we introduce Bayesian networks and describe two basic tasks common for both applications discussed in this paper: (1) building a Bayesian network model and (2) using the model to find a solution strategy.

1.1 Bayesian networks

Bayesian networks are probabilistic graphical models that are capable of modelling domains comprising uncertainty. They were introduced to the field of expert systems by Pearl [17] and Spiegelhalter & Knill-Jones [21]. The first applications were an expert system for electromyography Munin [2] and the Pathfinder system [7]. Since then Bayesian networks were successfully applied in several areas. Strength of graphical models is not only that they enable efficient uncertainty reasoning with hundreds of variables (e.g. using the method of Lauritzen & Spiegelhalter [13]), but also they help humans to understand better the modelled domain. This is mainly due to their comprehensible representation by use of directed acyclic graphs representing dependencies between domain variables.

* This work was supported by the Grant Agency of the Czech Republic through grant nr. 201/02/1269.

Bayesian network consists of an directed acyclic graph (DAG) $G = (V, E)$, to each node $i \in V$ corresponds one random variable X_i with a finite set $\mathbb{X}_i$ of mutually exclusive states and a conditional probability table (CPT) $P(X_i \mid (X_j)_{j \in pa(i)})$, where $pa(i)$ denotes the set of parents of node i in graph G. See Figure 1 for an example of Bayesian network.

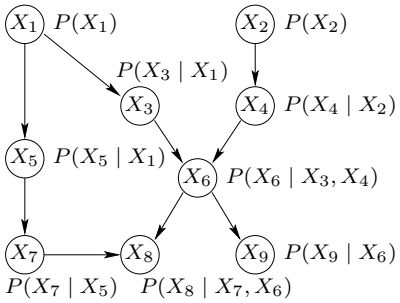

Fig. 1. An example of a Bayesian network

Bayesian network encodes qualitative and quantitative knowledge. Quantitative knowledge is represented by CPTs, while qualitative is encoded by use of a DAG. The DAG implies certain conditional independence relations between variables $(X_i)_{i \in V}$. A concept called *d-separation*, introduced by Pearl in [18], can be used to read the conditional independence statements from a DAG. X_i, X_j are *conditionally independent* given a set of variables $\mathcal{Y}$ if $P(X_i \mid \mathcal{Y}) = P(X_i \mid \mathcal{Y}, X_j)$. Two variables X_i and X_j are *d-separated* by $\mathcal{Y}$ if, for all trails [3] between nodes i and j there is an intermediate node k such that either

(1) edges do not meet head-to-head in k and $X_k \in \mathcal{Y}$, or
(2) edges meet head-to-head in k and neither X_k nor any of its descendants belongs to $\mathcal{Y}$.

It is required that all variables X_i, X_j d-separated by a set $\mathcal{Y}$ are conditionally independent given $\mathcal{Y}$ in a probability distribution P represented by the Bayesian network model. It is not difficult to show that the distribution satisfying the above property and having its CPTs equal to $P(X_i \mid (X_j)_{j \in pa(i)}), i \in V$ is unique and equals to the product of the CPTs, i.e.

$$P((X_i)_{i \in V}) = \prod_{i \in V} P(X_i \mid (X_j)_{j \in pa(i)}) \ .$$

For a detailed introduction to Bayesian networks we refer to [9].

[3] A trail in G is a sequence of nodes that forms a path in the undirected version of G, i.e. when the directions of arrows are ignored.

1.2 Building models

Three basic approaches can be used to construct a Bayesian network model:

- A model designer discuss domain experts who provide their expert knowledge of the modelled domain. Their knowledge is used to build the structure of the model and for specification of the values of CPTs.
- A database of records is collected. Then, a machine learning method is used to construct the model structure and to estimate the CPTs.
- Third approach is a combination of the previous two. For example, the model structure is designed consulting an expert, but the parameters of CPTs are learned from the collected data.

Even a brief review of different techniques that can be used to build the models is out of the scope of this paper. We refer an interested reader to [5]. In Section 2 we describe how a Bayesian network model used in an adaptive test can be constructed.

1.3 Building strategies using the models

A *strategy* describes steps that the user should perform in order to achieve a required goal. For example, a step can be: the user performs an action, the user makes an observation, or the user answers a question. Since outcomes of steps are uncertain each strategy must specify the next step the user should do for all possible combinations of outcomes of previous steps. Thus a strategy can be represented by a directed tree. It is convenient to define two types of nodes in the tree - *chance nodes* and *terminal nodes*. Each chance node corresponds to a single step of a strategy. Terminal nodes are leaves of the tree where the strategy terminates. One session corresponds to a path in the tree, i.e. to a sequence of steps, starting at the root of the tree and ending at a terminal node. In Figure 2 we present an example of an adaptive test consisting of two questions. Ovals denote chance nodes. Diamonds denote terminal nodes. Each chance node is labelled by the corresponding step. Every edge coming out from a chance node is labelled by an outcome of the step corresponding to that node. The strategy represented by the tree is: *If the answer to the first question X_2 is correct then the second question is X_3 otherwise the second question is X_1.*

Let $\mathcal{S}$ denote the set of all strategies admissible for a given problem and $\mathcal{L}(\mathbf{s})$ denote the set of all terminal nodes of a strategy $\mathbf{s} \in \mathcal{S}$. An evaluation function $f : \cup_{\mathbf{s}\in\mathcal{S}}\mathcal{L}(\mathbf{s}) \mapsto \mathbb{R}$ is defined. The goal is to minimise this function at the end of a session. The outcomes of the steps proposed in a strategy $\mathbf{s}$ are unknown, only the probabilities $P(\mathbf{e}_\ell)$ of terminating in a node $\ell \in \mathcal{L}$ can be computed from the domain model represented by a Bayesian network. Thus, using the expected value of the evaluation function defined for each strategy $\mathbf{s} \in \mathcal{S}$ as

$$E_f(\mathbf{s}) = \sum_{\ell\in\mathcal{L}(\mathbf{s})} P(\mathbf{e}_\ell) \cdot f(\mathbf{e}_\ell) \tag{1}$$

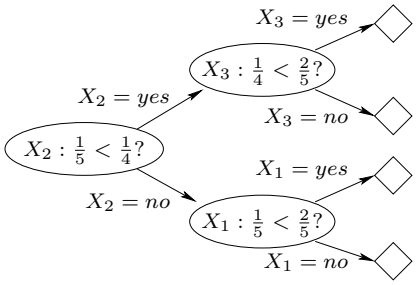

Fig. 2. Example of a strategy

we can formalise the task as a search for a strategy $\mathbf{s}^\star \in \mathcal{S}$ minimising the value of $E_f(\mathbf{s})$ from all $\mathbf{s} \in \mathcal{S}$. Due to the combinatorial explosion it is often impossible to find an optimal strategy. Instead different heuristic methods are used to find a reasonably good suboptimal strategies.

In Section 2 we apply this framework to adaptive testing and in Section 3 to troubleshooting. The two applications have different Bayesian network models, different sets of admissible strategies and different evaluation functions f. We will see that also the methods used to find suboptimal strategies are different.

2 Adaptive testing

Tests that are automatically tailored to the level of the individual examines are called *adaptive tests*. After each response on a question the system selects next question based on the answers of the previous questions. A simple example of adaptive test was presented in Figure 2. Since this approach requires computers for the test administration it is often referred to as *computerised adaptive testing* (CAT). For more information on CAT see [24] and [14].

2.1 Bayesian network model for adaptive testing

Almond & Mislevy [1] proposed to use graphical models for CAT. Their model consists of one *student model* and several *evidence models*, one for each task or question. Typically, a test designer specifies the tested skills $\mathcal{Y} = \{Y_1, \ldots, Y_k\}$ and a bank of questions $\mathcal{X} = \{X_1, \ldots, X_m\}$. The student model describes relations between student's skills, abilities, misconceptions. The knowledge about a student is expressed by use of a joint probability distribution $P(Y_1, \ldots, Y_k)$ defined on the variables of the student model.

Next we will describe learning process of a Bayesian network model used for testing basic operations with fractions introduced in [23].

First, a group of students from Aalborg University prepared paper tests that were given to students of Brønderslev High School. Four elementary skills (addition, subtraction, multiplication, and comparison), four operational skills (can-

celling, conversions between improper fractions and mixed numbers and vice-versa, common denominator), and abilities to apply operational skills to complex tasks were tested. The university students summarised the results as a list of data records. Several misconceptions were discovered and included as variables into the model.

Second, a model structure was learned using the PC-algorithm of Spirtes et al. [22], implemented in Hugin [8]. It provided a first insight into the relations between skills and misconceptions. Then a "domain expert" explained some relations with the help of hidden variables and introduced certain constraints on edges. Applying different constraints on the resulting model the final model was learned, again using the PC-algorithm. The final model was calibrated using the EM-algorithm [12].

In Figure 3 the final version of the student model is presented. Nodes in the first level (from bottom) correspond to the observed misconceptions, grey nodes in the second level to elementary skills, nodes with no fill in the second level correspond to operational skills, and shaded nodes in the third level to application skills. The top node corresponds to a hidden variable. See [23] where a more detailed description can be found.

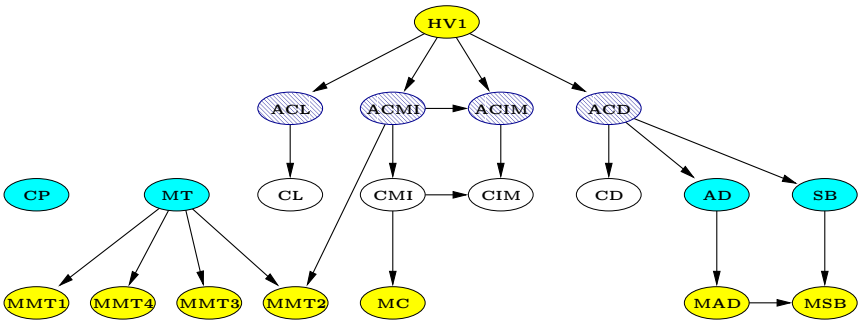

Fig. 3. Student model describing relations between skills and misconceptions.

For each question or task $X_j \in \mathcal{X}$ one evidence model is created by a domain expert. It describes the dependence of X_j on relevant skills from the student model. For every question the expert specifies which skills are related to the question. An example of a task is $\frac{1}{3} - \frac{1}{12} = \frac{4}{12} - \frac{1}{12} = \frac{3}{12} = \frac{1}{4}$. In order to be able to solve the task the student should have skills SB (subtraction) CL (cancelling), ACL (application of cancelling), CD (common denominator), ACD (application of common denominator), and should not have MSB (a misconception in subtraction). Thus, the relation between a variable T_j and related skills and misconceptions is described by a Boolean function. However, a student can make a mistake even if she has all abilities necessary to solve a given task and a correct answer does not necessarily mean that the student has all abilities. This

uncertainty was modelled by a conditional probability distributions $P(X_j \mid T_j)$ estimated from the collected data.

2.2 Building adaptive tests

Typically an adaptive test terminates after a given number of questions is answered or if sufficient information about the tested student is achieved. This defines the set of all possible test strategies $\mathcal{S}$.

Every examiner tends to maximise information about the student at the end of a test. A way to formalise this preference is to aim at a probability distribution $P(Y_1, \ldots, Y_k)$ minimising the Shannon entropy at the end of the test [3]. Entropy $H(P)$ of $P(Y_1, \ldots, Y_k)$ is defined as

$$H(P) = -\sum_{y_1,\ldots,y_k} P(Y_1 = y_1, \ldots, Y_k = y_k) \cdot \log P(Y_1 = y_1, \ldots, Y_k = y_k) \ .$$

In every terminal node ℓ of a test strategy $\mathbf{s}$ we will compute entropy of the conditional probability distribution given the evidence collected as far, i.e.

$$H(\mathbf{e}_\ell) = H(P(Y_1, \ldots, Y_k \mid \mathbf{e}_\ell)) \ .$$

Using substitution $f(\mathbf{e}_\ell) = H(\mathbf{e}_\ell)$ formula 1 can be written as

$$E_H(\mathbf{s}) = \sum_{\ell \in \mathcal{L}(\mathbf{s})} P(\mathbf{e}_\ell) \cdot H(\mathbf{e}_\ell) \ . \tag{2}$$

The goal is to find a test strategy $\mathbf{s} \in \mathcal{S}$ minimising the expected entropy $E_H(\mathbf{s})$.

In practice greedy heuristics are used to construct a myopically optimal test. A myopically optimal test is a test that consists of questions such that each question minimises the expected value of entropy after the question is answered.

2.3 Results and comparisons with other approaches

Classical approach used in educational and psychological testing since 1960's is item response theory (IRT) [15, 19]. Within this method the student is modelled by a single variable Θ. These models are suitable when the task is to grade students, but their application is problematic if more information about the tested student is required. In multidimensional IRT several variables are used to represent a student. The presented application of Bayesian network can be regarded as a generalisation of the multidimensional IRT. It brings two basic advantages:

- It can better reflect the student reasoning process and provides better insight into the modelled problem.
- Since the student model encode dependence between skills the adaptive tests are substantially shorter while the test precision is kept.

Results of experiments with adaptive test of basic operations with fractions were presented in [23]. Bayesian network used in adaptive tests provided good predictions of skills. In average more than 90% of skills were correctly predicted after seven questions were answered. In paper and pencil tests twenty questions were typically needed to get the same prediction quality.

See also [16] and [4] for results of experiments with Bayesian networks applied to adaptive diagnosis and tutoring.

3 Troubleshooting

Troubleshooting a man-made device is often quite a complex task. Therefore a system that uses evidence derived by performing repair actions or observations can substantially shorten the troubleshooting process [6].

3.1 Bayesian network model for troubleshooting

Within the approach presented in [10] a troubleshooting problem is modelled with a Bayesian network encoding relations among three types of variables: *faults* of the device $F \in \mathcal{F}$, *actions* $A \in \mathcal{A}$ - troubleshooting steps that may fix the problem, and *questions* $Q \in \mathcal{Q}$ - troubleshooting steps that help to identify the fault. Every action and question has a cost c assigned. We will use the simplified Light print example borrowed from [20] to illustrate the troubleshooting process.

Example 1 (Light print example). Suppose a printer prints a page that is too light. There can be dozens of possible printer faults in the case of a light print problem. Let us consider a simplified model with four possible faults: F_1 *Distribution problem*, F_2 *Defective toner*, F_3 *Corrupted dataflow*, and F_4 *Wrong driver setting*. Actions that may fix these faults can be A_1 *Remove, shake and reseat toner* with cost $c_1 = 5$, A_2 *Try another toner* with $c_2 = 15$, and A_3 *Cycle power* with $c_3 = 1$.

Generally, for each action an expert provides a table $P(A_i = yes \mid F_j)$. For example, action A_2 *Try another toner* fixes *Distribution problem* and *Defective toner* with the probability 0.9, i.e. $P(A_2 = yes \mid F_i) = 0.9, i = 1, 2$, but it does not fix *Wrong driver setting* at all, i.e. $P(A_2 = yes \mid F_4) = 0$. During the troubleshooting session it is often advisable to ask the user to answer some questions. The answers may help fix the problem faster by identifying the device fault. For instance if the answer to question Q_1 *Is the configuration page printed light?* is negative then the faults: *Distribution problem* and *Defective toner* are eliminated and the remaining faults are *Corrupt data flow* and *Wrong driver setting*. Generally, we have a table $P(Q_i = yes \mid F_j)$ for every question Q_i.

For many man-made devices it is reasonable to assume that actions and questions are *conditionally independent* given a fault. and that there is only one fault causing a device malfunction at a time. The second assumption is often referred to as the *single fault assumption*. The Bayesian network in Fig. 4 reflects both assumptions.

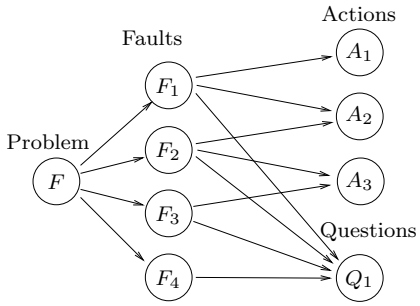

Fig. 4. Bayesian network model for the simplified light print example

3.2 Troubleshooting strategy

There are two ways for a troubleshooting strategy to terminate, either by fixing the problem or by giving up. Thereupon two types of *terminal nodes* are defined:

(1) *Success terminal nodes* correspond to fixing the problem.
(2) *Failure terminal nodes* correspond to giving up the troubleshooting.

Figure 5 provides an example of a troubleshooting strategy. The success terminal nodes are shaded while failure terminal nodes are not.

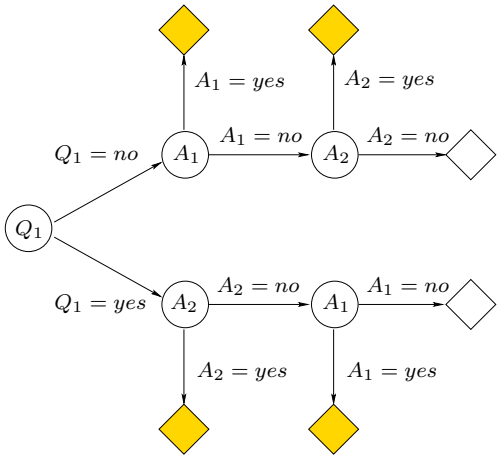

Fig. 5. Troubleshooting strategy

A function $CR(\mathbf{e}_\ell)$ that corresponds to the cost of repair of the device is defined. It has two components. The first component $t(\mathbf{e}_\ell)$ is the total cost of

actions and questions performed while getting to state $\mathbf{e}_\ell$ corresponding to node ℓ in the tree of a strategy $\mathbf{s}$. The second component is the penalty function $c(\mathbf{e}_\ell)$ that applies for every terminal node ℓ. The penalty is defined to be zero if the problem is fixed, i.e. for all success terminal nodes. In the failure terminal nodes the penalty is a positive number that may be interpreted as a cost of calling service. Thus $CR(\mathbf{e}_\ell) = t(\mathbf{e}_\ell) + c(\mathbf{e}_\ell)$. Substituting $f(\mathbf{e}_\ell) = CR(\mathbf{e}_\ell)$ to formula 1 we get a criteria called *expected cost of repair*.

$$E_{CR}(\mathbf{s}) = \sum_{\ell \in \mathcal{L}(\mathbf{s})} P(\mathbf{e}_\ell) \cdot (\ t(\mathbf{e}_\ell) + c(\mathbf{e}_\ell) \ ) \quad . \tag{3}$$

Troubleshooting task is to find a strategy $\mathbf{s} \in \mathcal{S}$ minimising $E_{CR}(\mathbf{s})$.

A solution of the *troubleshooting task* can be easily found in the case of *independent actions*, that is in the situations when (1) every action fixes just one fault and (2) all actions are pairwise independent. In the case of *independent actions* with *single fault assumption* it suffices to order actions decreasingly according to the ratio $P(A = yes)/c_A$, see [11]. In [20] it was shown that if some actions fix more than one fault the *troubleshooting task* becomes NP-hard. Therefore methods that provide reasonably good troubleshooting strategies in real-time are necessary.

3.3 Results

The troubleshooter developed within the SACSO project exploits Bayesian network models. Since the models have the structure of the Naïve Bayes model, many probability propagations can be performed whenever a new troubleshooting step has to be chosen. The approach exploits heuristics based on the $P(A = yes)/c_A$ ratio and extends the myopic approach proposed in [6] for troubleshooting that includes observations. In [10] the strategies obtained using the SACSO approach were compared with the optimal solutions. The E_{CR} values of the strategies provided by the SACSO troubleshooter were very close to the optimal values. The troubleshooter is available on the market under the name DezisionWorks. It is sold by a Danish company Dezide, see `http://www.dezide.com/`.

References

1. R. G. Almond and R. J. Mislevy. Graphical models and computerized adaptive testing. *Applied Psychological Measurement*, 23(3):223–237, 1999.
2. S. Andreassen, F. V. Jensen, S. K. Andersen, B. Falck, U. Kjærulff, M. Woldbye, A. R. Sørensen, A. Rosenfalck, and F. Jensen. MUNIN — an expert EMG assistant. In J. E. Desmedt, editor, *Computer-Aided Electromyography and Expert Systems*, chapter 21. Elsevier Science Publishers, Amsterdam, 1989.
3. M. Ben-Bassat. Myopic policies in sequential classification. *IEEE Transactions on Computers*, 27(2):170–174, 1978.
4. C. Conati, A. S. Gertner, K. VanLehn, and M. J. Druzdzel. On-line student modeling for coached problem solving using Bayesian networks. In A. Jameson, C. Paris, and C. Tasso, editors, *Proc. of the Sixth Int. Conf. on User Modeling (UM97), Chia Laguna, Sardinia, Italy*. Springer Verlag, 1997.

5. R. G. Cowell, A. P. Dawid, S. L. Lauritzen, and D. J. Spiegelhalter. *Probabilistic Networks and Expert Systems*. Statistics for Engineering and Information Science. Springer Verlag, 1999.
6. D. Heckerman, J. S. Breese, and K. Rommelse. Decision-theoretic troubleshooting. *Communications of the ACM*, 38(3):49–57, 1995.
7. D. Heckerman, E. Horwitz, and B. Nathwani. Towards normative expert systems: Part I, the Pathfinder project. *Methods of Information in Medicine*, 31:90–105, 1992.
8. Hugin. Hugin Explorer, ver. 6.0. Computer software, 2002. `http://www.hugin.com`.
9. F. V. Jensen. *Bayesian networks and decision graphs*. Statistics for Engineering and Infromation Science. Springer Verlag, New York, Berlin, Heidelberg, 2001.
10. F. V. Jensen, U. Kjærulff, B. Kristiansen, H. Langseth, C. Skaanning, J. Vomlel, and M. Vomlelová. The SACSO methodology for troubleshooting complex systems. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 15:321–333, 2001.
11. J. Kalagnanam and M. Henrion. A comparison of decision analysis and expert rules for sequential analysis. In P. Besnard and S. Hanks, editors, *The Fourth Conference on Uncertainty in Artificial Intelligence*, pages 271–281, New York, 1988.
12. S. L. Lauritzen. The EM-algorithm for graphical association models with missing data. *Computational Statistics and Data Analysis*, 1:191–201, 1995.
13. S. L. Lauritzen and D. J. Spiegelhalter. 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.
14. W. J. V. D. Linden and C. A. W. Glas, editors. *Computerized Adaptive Testing: Theory and Practice*. Kluwer Academic Publishers, 2000.
15. F. M. Lord. *A theory of test scores*. Number 7 in Psychometric Monographs. Psychometric Society, 1952.
16. E. Millán and J. L. Pérez-de-la-Cruz. A bayesian diagnostic algorithm for student modeling and its evaluation. *User modeling*, 2002.
17. J. Pearl. Reverend Bayes on inference engines: a distributed hierarchical approach. In *Proceedings, AAAI National Conference on AI,Pittsburgh, PA*, pages 133–136, August 1982.
18. J. Pearl. Fusion, propagation and structuring in belief networks. *Artificial Intelligence*, 29(3):241–288, 1986.
19. G. Rasch. Probabilistic models for some intelligence and attainment tests. Technical report, Danish Institute for Educational Research, Copenhagen, 1960.
20. M. Sochorová and J. Vomlel. Troubleshooting: NP-hardness and solution methods. In *Proceedings of the Fifth Workshop on Uncertainty Processing WUPES'2000*, Jindřichův Hradec, Czech Republic, 20–24th June 2000.
21. D. J. Spiegelhalter and R. P. Knill-Jones. Statistical and knowledge-based approaches to clinical decision-support systems. *Journal of the Royal Statistical Society, Series A*, (147):35–77, 1984.
22. P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction, and Search*. Number 81 in Lecture Notes in Statistics. Springer Verlag, 1993.
23. J. Vomlel. Bayesian networks in educational testing. In *Proceedings of First European Workshop on Probabilistic Graphical Models (PGM'02)*, Cuenca, Spain, November 6-8 2002.
24. H. Wainer, D. Thissen, and R. J. Mislevy. *Computerized adaptive testing : a primer*. Mahwah, N.J. : Lawrence Erlbaum Associates, second edition, 2000.