Computers & Education 55 (2010) 1663–1683

Contents lists available at ScienceDirect

Computers & Education

journal homepage: www.elsevier.com/locate/compedu

Bayesian networks for student model engineering

Eva Millán [a,*], Tomasz Loboda [b], Jose Luis Pérez-de-la-Cruz [a]

[a] ESTI Informática, Universidad de Málaga, Málaga 29071, Spain
[b] School of Information Sciences, University of Pittsburgh, Pittsburgh, PA 15260, US

ARTICLE INFO

Article history:
Received 24 November 2009
Received in revised form
19 July 2010
Accepted 19 July 2010

Keywords:
Bayesian networks
User modeling
Student modeling
User model
Student model

ABSTRACT

Bayesian networks are graphical modeling tools that have been proven very powerful in a variety of application contexts. The purpose of this paper is to provide education practitioners with the background and examples needed to understand Bayesian networks and use them to design and implement student models. The student model is the key component of any adaptive tutoring system, as it stores all the information about the student (for example, knowledge, interest, learning styles, etc.) so the tutoring system can use this information to provide personalized instruction. Basic and advanced concepts and techniques are introduced and applied in the context of typical student modeling problems. A repertoire of models of varying complexity is discussed. To illustrate the proposed methodology a Bayesian Student Model for the Simplex algorithm is developed.

1. Introduction

Present-day instructional technology is concerned with the delivery of content that is flexible and adaptable to each and every learner. In this sense, student modeling plays a central role in many educational computer systems and has important implications for e-learning systems. Student modeling has been called "the key to individualized knowledge-based instruction" (Greer & McCalla, 1994).

Many approaches to modeling have been tried in the area of student modeling (Brusilovsky & Millán, 2007). Among them, Bayesian networks (BNs) have attracted a lot of attention from theorists and system developers due to their sound mathematical foundations and also for a natural way of representing uncertainty using probabilities (Jameson, 1996; Liu, 2008). In the current paper, we provide education practitioners with an introduction to using BNs as a practical tool for student modeling.

We start by discussing the basic concepts of student modeling (Section 2). Then, we present a short tutorial on BNs, with the emphasis on the modeling process (Section 3). After that, we focus on the application of BNs in the context of student modeling problems (Section 4). We finish with some concluding remarks (Section 6).

2. Student modeling

2.1. Student model

A student model is one of the components of the traditional architecture for Intelligent Tutoring Systems (ITSs) (Wenger, 1987). This model keeps track of the progress the student makes while they interact with the learning material. Quoting Self (Self, 1999), "a student model is what enables a system to care about a student". An ITS uses the information stored in a student model to tailor the way it interacts with a student. For instance, it could attempt to correct misconceptions a student might have, provide personalized feedback, suggest learning a particular item suitable given the current level of knowledge of the student, etc.

* Corresponding author. Tel.: +34 952131397; fax: +34 952132814.
E-mail address: eva@lcc.uma.es (E. Millán).

0360-1315/$ – see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.compedu.2010.07.010

2.2. Modeling: objects and techniques

An ITS assigns a separate copy of a student model to each student. The designer of an ITS defines a prototype those student models are to be based upon. Two considerations should be taken into account when forging those prototypes:

- How will the student model be initialized and updated?

Many researchers have stressed the inherent difficulty of student model construction and initialization (e.g.,Beck 2007; Self, 1988; Zukerman & Albrecht, 2001). To guarantee a reasonable fidelity of the representation of the world a student model provides, it is necessary to set a lower bound on its complexity.

- How will the student model be used?
  The intended use of a given student model would set an upper bound on its complexity. To cite Self (1988), "it is not essential that ITSs possess precise student models, containing detailed representations of all the component mentioned above, in order to tutor students satisfactorily (…) there is no practical benefit to be gained from incorporating in our student models features which the tutoring component makes no use of".

Those considerations determine the designer's decisions concerning the *level of detail* and *items* about which the system maintains information or beliefs. With respect to the level of detail, the simplest models are scalar models, that store merely a single number that summarizes the state of the student's current learning progress. The most complicated models are semantic-network-like models, having hundreds of nodes that allow for modeling of several levels of detail. With respect to the intended use of a student model, traditional models have mainly represented the knowledge that the system attributes to a student at a given time. This knowledge can be declarative (e.g., propositions learned) procedural (e.g., skills mastered), or a mixture of both (e.g., rules learned). There have been many proposals to enrich the vector of modeled variables with the learner's *attributes* and *aptitudes* that can be classified as (Self, 1994):

- *Cognitive*. For example, the system believes that the student has a "good visual analogical intelligence."
- *Conative*. Desires, intentions, and cognitive and learning styles, etc. For example, the system believes that the student is "reflective" rather than "impulsive."
- *Affective*. Attributes related to values, e.g., motivation, self-concept, etc. A good deal of current research is aimed to model learner's emotions.

The information about the learner and their progress is stored in a knowledge representation model. Lately, there has been an increasing interest in using BNs in user modeling, and, in particular, in student modeling. There are important reasons for this increase. First, BNs is a theoretically sound framework. This guarantees that any unanticipated behavior of the model is a result of the designer error. Second, today's computer machinery is much more powerful than a few years ago. That renders inference algorithms that BNs rely on more efficient and allows for solving networks of considerable size. An additional factor facilitating the adoption of BNs (and somewhat related to the second reason mentioned above) is the presence of capable and robust Bayesian libraries (e.g., SMILE). Those libraries can be easily integrated into the existing or new student modeling applications. We think BNs are one of the best options for building student models.

2.3. Types of student models

Several widely used types of student models are:

- *Overlay model* (Carbonell, 1970). In this approach, the student's knowledge is considered to be a proper subset of the knowledge of the entire domain. Differences in the behavior of the student as compared to the behavior of someone with perfect knowledge are treated as an indication of knowledge gaps. This model works reasonably well whenever the main goal of the tutoring system is to transmit knowledge from the system to the student. The major difficulty is that it does not consider the fact that the student can have erroneous beliefs, and therefore the system will be incapable of dealing with this sort of mistake.
- *Differential model* (Burton & Brown, 1982). It is a variation on the overlay model. In the differential model, the domain knowledge to be learned by the student is decomposed into necessary and unnecessary (or optional). This model can be thought of as the overlay model, but defined over a subset of the domain knowledge.
- *Perturbation model* (Brown & Burton, 1978). In this model, the student's knowledge is dichotomized into "correct" and "incorrect". It is usually built upon the domain knowledge augmented with the more common mistakes students make. The student model is then the overlay model over an augmented set of knowledge items, which includes both correct and incorrect knowledge propositions. Incorrect knowledge is further divided into misconceptions and bugs. The collection of mistakes included in a perturbation model is usually called *bug library* and can be built either by empirical analysis of mistakes (enumeration) or by generating mistakes from a set of common misconceptions (generative techniques). The perturbation model provides better explanations of the student's behavior, while being more costly to build and maintain.
- *Constraint-based model* (Mayo & Mitrovic, 2001; Mitrovic, 2003; Mitrovic, Koedinger, & Martin, 2003, chap. 42, pp. 313–322; Ohlsson, 1992). The most common type of modeled knowledge is propositional knowledge (e.g., "primates are mammals"). However, there are alternative representations. For example, in constraint-based models the domain knowledge is represented by a set of constraints over the problem state. This set of constraints identifies correct solutions and the student model is an overlay model over this set. The advantage of this model is that unless a solution violates at least one constraint, it is considered correct. That allows the student to employ different ways of problem-solving, including those not foreseen by the student model engineers.

- We must distinguish between two different types of student modeling, *knowledge tracing and* model tracing (Anderson, Corbett, Koedinger, & Pelletier, 1995; Baker, Corbett, & Aleven, 2008; Conati, Gertner, & Vanlehn, 2002; Conati, Gertner, & VanLehn, Druzdzel, 1997; Corbett & Anderson, 1992; Corbett, Anderson, & O'Brien, 1995; Kasurinen & Nikula, 2009; Pardos & Heffernan, 2010). Knowledge tracing attempts to determine what the student knows, including misconceptions they might have. It is an approach to assessment. Model tracing attempts to understand how the student solves a given problem. It is an approach to plan recognition. Model tracing is especially useful in systems that are intended to provide guidance when the student reaches an impasse. Knowledge tracing is also useful as an evaluation tool and a pedagogical decisions aid (e.g. which piece of learning material to present next).

BNs can be used to implement all of the above approaches.

3. Bayesian networks

There are many good introductory publications to BNs (see, e.g., Neapolitan, 1990; Charniak, 1991). In this section, we briefly present main concepts necessary to understand the remainder of the paper. We formally define BNs, explain the conditional independence assumption, and discuss the inference process. Familiarity with the basics of the probability theory is assumed.

3.1. Definition

A Bayesian network is defined as (Neapolitan, 1990):

- A set $V = X_1, \ldots, X_n$ of random variables, such that each variable X_i takes a set of *exhaustive and mutually exclusive* values. These variables will be represented as nodes in the network.
- A set E of probabilistic relationships between the variables. These relationships will be represented as arcs (or links) in the network.
- A joint probability distribution P, defined in V (as a reminder, $P(x_1, \ldots, x_n)$ denotes the probability that $X_1 = x_1$, and … and $X_n = x_n$),

such that:

- The graph $G = (V, E)$ is a directed acyclic graph (DAG),
- The set (V, P) satisfies the conditional independence assumption.

If all the values of the joint probability distribution $P(x_1, \ldots, x_n)$ are known, then one can perform any kind of inference. However, the elicitation of the joint probability distribution is a task that requires a number of parameters that is exponential in the number of variables. If there are n binary variables, then 2^n values are needed to fully specify $P(x_1, \ldots, x_n)$. Moreover, the computation of posterior probability distributions from $P(x_1, \ldots, x_n)$ is a computationally complex process.

As we show in the next subsection, under the conditional independence assumption that BNs make use of, the number of required parameters can be significantly reduced. The computation of posterior probability distributions is also easier to perform.

3.1.1. Conditional independence assumption

A DAG $G = (V, E)$ and a joint probability distribution P are said to meet the conditional independence assumption if and only if:

- For every variable X &;ie; V and for every variable Y &;ie; $V\backslash\{X, \gamma, \pi\}$, X is independent of Y given π,

where π is the set of direct predecessors (or *parents*) of X and γ is the set of direct successors (or *children*) of X.

The conditional independence assumption is very important in the Bayesian framework. Under this assumption, the joint probability can be computed by multiplying the conditional probabilities of every variable given its parents, i.e. (Pearl, 1988):

$$P(X_1, \ldots, X_n) = \prod_i P(X_i|\pi_i) \ .$$

Therefore, if the conditional independence assumption is satisfied, it is enough to provide:

- For every root node (a node without parents), a prior probability distribution,
- For every other node, a conditional probability distribution associated with every of its parents.

To illustrate those savings, let us consider a BN with binary variables from Fig. 1. For the joint probability, we need to provide $2^5 = 32$ numbers (for all combinations of values of the five variables). But if conditional independence is assumed, we only need to provide nine parameters (one prior and eight conditional probability distributions). In general, in a network with n binary variables, the number of parameters needed is $2(n - 1) + 1$ instead of 2^n. In other words, the number of parameters becomes linear in the number of nodes, instead of

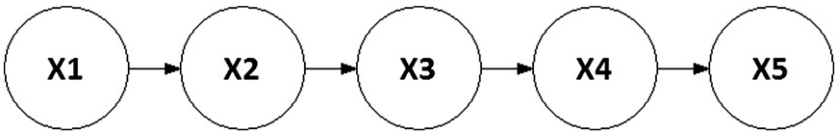

Fig. 1. An example Bayesian network.

being exponential. An additional advantage of conditional probabilities is that they are easier to elicit from domain experts. In Pearl's words (Pearl, 1985):

> "Human performance, by contrast, exhibits an opposite complexity ordering: probabilistic judgements on a small number of propositions (specially conditional statements) are issued swiftly and reliably, while judging the likelihood of a conjuction of many propositions is done with great degree of difficulty and hesitancy".

We elaborate on conditional independence in Section 3.4.

According to Pearl (Pearl, 1988), what is Bayesian about Bayesian networks is that they comply with the two defining attributes of the Bayesian school:

1. Willingness to accept subjective opinions as expedient substitute for raw data
2. Adherence to Bayes conditionalization as the primary mechanism for updating beliefs.

In the next section, we will explain how Bayesian Networks are used to perform approximate reasoning.

3.2. Inference

Once a BN is created, it can be used to reason about the situation it models. Inference in the Bayesian framework entails the computation of the probability distribution over all variables given the available *evidence* (or set of *observations*). This process is sometimes referred to as *beliefs update*. After beliefs update, a *posterior probability distribution* is associated with each variable. That distribution reflects the influence of evidence.

BNs allow for two kinds of reasoning: *diagnostic* and *predictive*. Diagnosis is a task of identifying the most likely causes given a set of observations. Observations in that context are sometimes referred to as *symptoms* or *faults*. A good example of this kind of reasoning is that of medical diagnosis, where the focus is on identifying diseases a patient may be suffering from and ordering them from the most likely ones to the least likely ones given test results. Another good example is that of troubleshooting. A technician may be interested in finding out the most likely cause of a machine's malfunction given a set of symptoms (e.g., smoke coming out of it). Prediction (or *forecasting*), on the other hand, attempts to identify the most likely event occurrence given a set of observations. Diagnosis looks at the past and present to reason about the present while prediction looks at the past and present to reason about the future.

In a BN, any variable can be either a source of information (if its value is observed) or object of inference (given the set of values that other variables in the network have taken). Reasoning will then be diagnostic or predictive depending on what evidence is available.

Student modeling may be both diagnostic and predictive. It is diagnostic when the task is to assess the current level of the student's understanding of the learning material. Subsequent grade assignment could be conditional on that assessment. Student modeling is predictive when the focus is on assessing the student's performance on a piece of material yet to be learnt. As such, it may be used to guide selection of the problem to be presented to the student next. The distinction between these two types of reasoning may blend in many real-life applications and it is often difficult to draw a line between them.

Inference algorithms are responsible for computing posterior probability distributions for all variables. The problem with this computation is that probabilistic inference in some classes of BNs is NP-hard (Cooper, 1990). Many algorithms have been developed (see, e.g., Lauritzen & Spiegelhalter, 1988; Pearl, 1988). Most of them solve reasonably large and complicated BNs in an acceptable time. Whenever efficiency is preferred over accuracy, or the structure of models prohibits exact solutions, approximate algorithms are employed. Evidence propagation algorithms are beyond the scope of this discussion.

A number of software tools, such as GeNIe & SMILE[1] or HUGIN[2], allow for easy graphical construction of Bayesian models and provide libraries of belief update algorithms. All models presented in this paper have been created in GeNIe, developed by Decision Systems Laboratory of the University of Pittsburgh.

3.3. Modeling

Modeling can be defined as the task of building a representation of a real world scenario (for a more philosophical vantage see, e.g., Weisberg, 2007). In this section we construct an example Bayesian model. Below is the description of an example situation we use. This problem is a mutation of the example discussed in Druzdzel and Henrion, 1990.

John was invited for dinner by his friends, Paul and Susan. Even though he was feeling perfectly fine the whole day, over appetizers he suddenly starts sneezing. John thinks that he might have caught cold, but that it can also be the case that he is having rhinitis, since he is allergic to several things. John notices that most of his friends' furniture have scratches. John thinks that they might have a cat. That would explain the reason he is sneezing – he is allergic to cats. John asks Susan about it and learns that indeed they have a cat. Therefore, John does not worry about having caught cold.

The sneezing example is of a diagnostic nature; John is trying to find out why he is sneezing.

This simple example will help us to get an intuition about the BN-related concepts. Here, we use Bayesian models to encode that representation. Therefore, to develop the model we need to define variables, connections between them (causal relationships), and parameters (probability distributions).

Before attempting model building it is imperative to gain good understanding of the domain of interest. Examples of questions that can help in gaining more insight include:

[1] GeNIe (Graphical Network Interface) and SMILE (Structural Modeling, Inference, and Learning Engine) are available at http://genie.sis.pitt.edu.
[2] HUGIN (Handling Uncertainty In General Inference Networks) is available at http://www.hugin.com.

- How much expertise do I have in the domain?
- Should I contact someone who knows more (a domain expert)?
- the problem decomposable into smaller chunks (topics)?
- Can those chunks be organized in a hierarchy?
- there any ontologies already available for the domain?
- Am I comfortable with assumptions those ontologies make?

Very often a model engineer working towards building a student model may need input from a teacher in a particular area or course.

For some domains a formal description in a form of ontology is available. It is often the case that several competing ontologies exist. These ontologies may be used for inspection and can be refined to create a basis for model construction. An example of a domain with well defined ontologies is the domain of computer programming.

A natural next step is to analyze and understand the problem itself. Examples of questions pertinent to this step are:

- How big is the problem?
- Can it be decomposed into subproblems that can be taken on separately?
- Do any of the subproblems look similar and can be considered instances of the same class of subproblems?

In the sneezing example, the problem is that John started sneezing and he wants to find out why. It is a rather simple problem and it does not seem to be decomposable into subproblems. In the domain of student modeling, a problem could be to capture the student's level of understanding of the domain concepts. In that case, each concept could be modeled separately. The model would consist of a number of submodels, possibly varying in similarity.

In the following subsections we define variables, connections between them, and model parameters for the sneezing example.

3.3.1. Variables

Modeling should start with identifying variables within the domain that are relevant to the problem. It is important to focus only on the clearly relevant ones. The following questions may help at this stage:

- What is the problem being modeled?
- What are its possible causes?
- Which other factors can make problems or causes happen, or prevent them from happening?
- Which evidence could be available to support causes, problems or factors?

In our case, the problem is that John is Sneezing. Possible causes are that he has Cold, or Rhinitis. Rhinitis might be caused by their friends having a Cat and John being allergic (Allergies) to cats. Evidence supporting that his friends have a cat is that furniture has Scratches. The above pieces of information are relevant. Example of irrelevant information is that of John's shoe size.

Variables can be divided into four classes with respect to the role they play in a model. We describe them below and summarize in Table 1.

3.3.1.1. Targets. These variables are used to model objects of interest, something we would like to reason about. They are also called *faults*, especially in the context of technical diagnosis. Target variables model phenomena that are usually *latent* (or not observable). That means that it is not possible to measure them directly. In the sneezing example John thinks about two alternatives: either he caught Cold or has Allergies. Both are examples of target variables, since John is interested in knowing more about them (the state they are in). In medical diagnosis, diseases would be modeled as target variables. An example of a target variable in a student model could be the level of understanding of a concept. It is a variable of interest, but it is measured indirectly, e.g. with a test or an exam.

3.3.1.2. Observations. Observation variables are used to model observable phenomena, usually providing information about target variables. They are also called *evidence variables* or *tests*. In the sneezing example, John thinks he is fine until he starts sneezing. Not until he makes an observation of himself sneezing is he wondering about the cause of it. Sneezing would be an observation variable. Another one would be Scratches, because John makes that observation and uses it to reason about a cat being in the house. In medical diagnosis, symptoms and test results would be modeled as observations. In student modeling, actions of the student, let it be user interface events (e.g., mouse clicks) or answers to specific assessment questions, are modeled as observation variables.

3.3.1.3. Factors. These variables are used to model other sources of influence affecting target variables. They are also called *context* or *background knowledge* variables. In the sneezing example, the Season could be a factor affecting Cold. A person is more likely to catch cold in the fall or spring than in the summer or winter.

Table 1
Classes of variables appearing in Bayesian network.

Class	Short description
Target	Models the object of interest; cannot be measured directly
Observation	Models ways of measuring target variables; can be measured directly
Factor	Models phenomena affecting other variables in the model
Promoter	The affected variable is more likely when this factor are present
Inhibitor	The affected variable is less likely when this factor are present
Required	This factor needs to be in action for the affected variable to happen
Barring	This factor needs not to be in action for the affected variable to happen
Auxiliary	Used only for modeling convenience (e.g., to simplify the network structure)

Factors can be divided into four categories with respect to the type of influence on the affecting variable.

- *Promoters*. These factors make the affecting variable more likely when they themselves become more likely (positive correlation). For example, smoking might increase the odds of having lung cancer.
- *Inhibitors*. These factors make the affecting variable less likely when they themselves become more likely (negative correlation). For example, participating in sports might decrease your odds of getting sick.
- *Required*. These factors are required to be in effect in order for the affecting variable to be considered probable. For example, a specific bacteria population growth requires a temperature to be above a certain threshold. If the temperature is below it, growth is impossible.
- *Barring*. When these factors are in effect they drive the probability of the affecting variable to zero. For example, receiving smallpox vaccine at early age prevents from suffering from that illness later in life. A barring factor can be considered as an inhibitor (to the extreme).

It is a mistake to include too much of background knowledge in the model. That makes a model less transparent. Simple models are easy to understand and evaluate. That does not mean one should sacrifice expressive power of the model by limiting the number of variables just for the sake of keeping things simple. A variable should only be included in the model if it provides pertinent information. If there are doubts about this, it usually can be left out. Ultimately, all things are connected somehow and a model cannot account for all of them.

3.3.1.4. Auxiliary. These variables are used for modeling convenience only. For example, if a node has many parents (recall that the number of parameters for a node needed is exponential in the number of parents of such node), intermediate auxiliary nodes can be used to group parents. In this way, the network structure is simplified and the number of parameters needed is smaller.

3.3.2. States and values

Depending on the nature of the phenomena being measured or properties of our measuring device, variables can be divided into *discrete* and *continuous*. A discrete variable can take on a finite set of values. These values, also called *states*, are mutually exclusive and exhaustive. An example of a discrete variable is Day-of-the-week. It has seven states corresponding to each day of the week. In the sneezing example, Allergy (that could be considered as continuous variable), can be modeled as discrete with four states: no, mild, medium, and severe. Sometimes a state Other is included to capture all "not-modeled" states, thus making the set of states a variable can take on exhaustive.

If a discrete variable has only two states it is called *binary*. An example of a binary variable is Gender; someone can be either a male or a female. It is very common for target variables (faults) to have present/absent states and observation variables (tests) to have positive/negative states.

A continuous variable can take on any value from a given range. Examples of continuous variables are temperature or time. It is possible to represent a phenomenon continuous in nature, like temperature, using a discrete variable. In order to do that, continuous measurements need to be *discretized*. That is done by projecting the continuous scale of measurement onto a finite set of ranges (so called *bins*). Values falling in the same bin will be treated as the same state. An example of discretization is modeling of temperature with a variable with three states: low, medium, and high.

In the sneezing example, we will consider all variables to be binary, so the possible values for each variable are present or absent. In more sophisticated problems and applications, we could use discrete variables with more states (for example, Sneezing could take one of three values: minor, moderate, severe) or continuous variables (e.g., a Body-temperature node would take values ranging from 35.8 to 42.5). Of course the more values a node has, the more complicated the model gets, so when deciding about the number of states an engineer should think about the degree of granularity that is necessary in a given application.

It is sometimes difficult to decide whether certain real world phenomena should enter the model as separate variables or as states of the same variable. As mentioned earlier, the set of values of each variable should be mutually exclusive. Is it possible for John from the sneezing example to have Rhinitis and Cold simultaneously? If the answer is affirmative, then each of these objects of the real world should be modeled as a variable. If the answer is negative, then the set {Rhinitis, Cold} is exclusive, and they should be modeled as two states of the same variable, say, Disease.

On the other hand, the set of values of each variable should be exhaustive. A common practice to ensure that this property is satisfied is to add a state Other to the variable. That state models all cases not explicitly accounted for by the other states.

3.3.3. Structure

After defining variables, next step in building a model is defining its structure. This is done by connecting variables with *arcs* (also called *links*). In Bayesian networks, arcs are directed. Changing the direction of an arc changes its meaning. When two nodes in a graph become connected with a directed arc, we refer to one of the variables as a *parent* (the antecedent) and to the other one as a *child* (the successor). (Fig. 2).

The sneezing example model could be a result of the following reasoning sequence (Section 3.3.3). First, John starts Sneezing (observation). Possible causes (targets) of Sneezing are Cold and Rhinitis. Next, John notices Scratches on furniture (observation). Then, he realizes those might be made by a Cat (target). He realizes that Sneezing could be due to Rhinitis, which in turn could be due to his Allergies (target).

Usually, the presence of an arc from a variable X to a variable Y denotes that X is a direct cause of Y. In the sneezing example, the presence of a Cat is a cause of Scratches, hence the arc between these two variables. Borrowing from Pearl (2000), p. 21, "the ubiquity of DAG models in statistical and AI applications stems (often unwittingly) primarily from their causal interpretation". On the other hand, the absence of an arc between X and Y means that X is not a direct cause of Y. In the sneezing example, a Cat is not a cause of Cold, hence those two variables remain unconnected.

From a mathematical or abstract point of view, BNs do not enforce the causal arc direction. However, there are important advantages of building causal BNs. Causal models are more meaningful for humans. Evidence from experimental psychology and introspection (Kahneman, Slovic, & Tversky, 1982) shows, that people store information in the form of causal associations rather than probabilistic judgments. Moreover, "causal relationships are more stable than probabilistic relationships" (Pearl, 2000, p. 25). Causal relationships are typically independent of one another and the modification of one of them does not affect the others. To sum up, "the patterns of

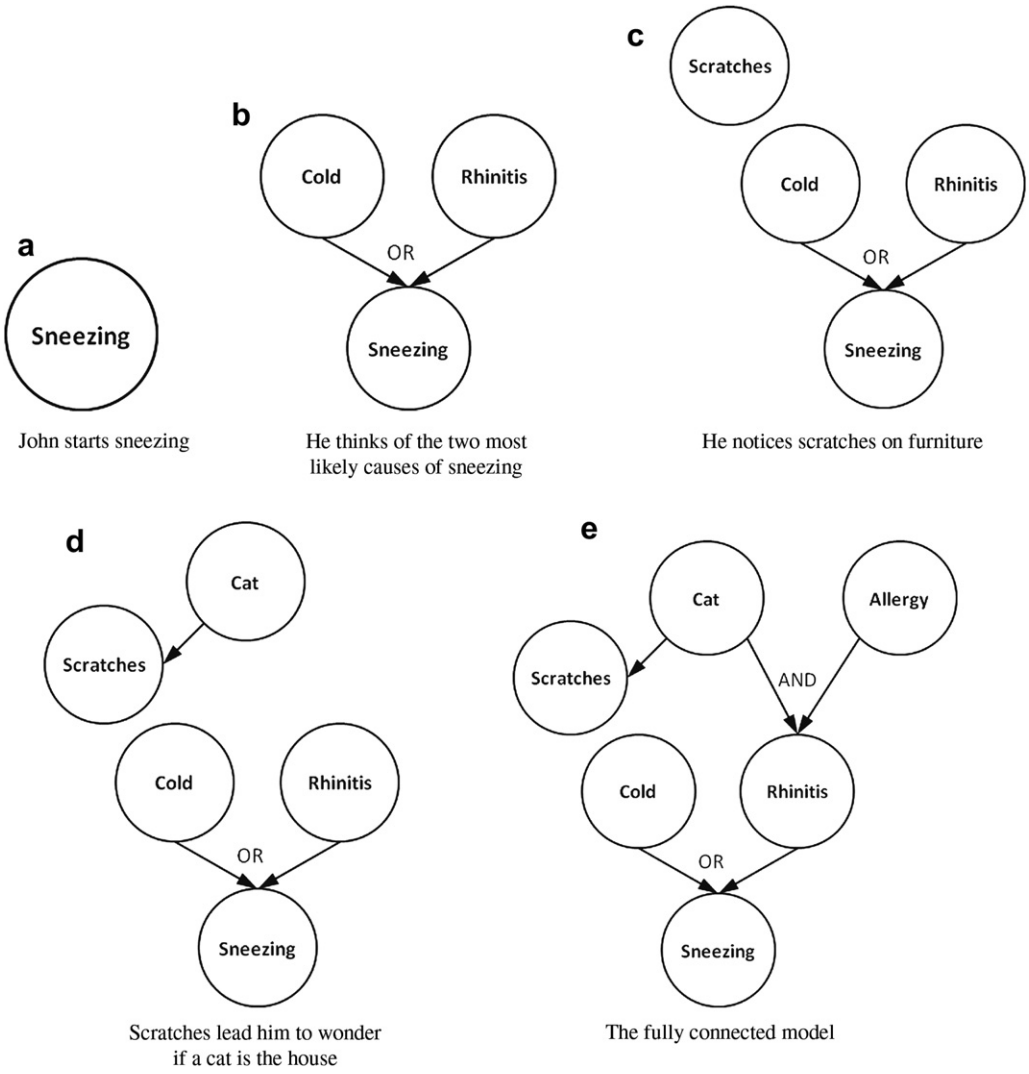

Fig. 2. A step-by-step modeling of the sneezing example.

independencies portrayed in a DAG are typical of causal organizations and some of these patterns can only be given meaningful interpretation in terms of causation" (Pearl, 1988).

An often proposed (explicitly or implicitly) alternative to the causal arc direction is the *diagnostic* direction (Fig. 3). Diagnostic direction, represents the relationship between different pieces in a reasoner's knowledge. For example, the reasoner could say "I have found out the result of the test *T*, therefore my belief about the fault *F* has changed". When using this direction, the observation variable is a parent of the target variable. In the light of the above considerations, this approach should be applied with care.

No matter the arc direction, it is possible to reverse any or all the arcs in an existing model. In fact, for any BN *A* there is another BN *B* such that all links in *A* are present in *B* but in the opposite direction (Druzdzel & van Leijen, 2001). However, the arc reversal process introduces additional links which are usually difficult to understand. Fig. 4 shows the sneezing example BN with all arcs reversed. Influences that characterize the interaction of variables from the model, while clearly visible in the original model, are arguably much less clear in the reversed model. For example, the influence the state of Rhinitis has on the state of Schatches.

Above, we assumed that the structure of the BN is defined by a human expert. An alternative approach is to infer the structure from a set of recorded cases. There are many machine learning techniques capable of defining the structure (and parameters) of a BN given a data set. We will elaborate on this topic in Section 4.4.

3.3.4. Parameters

The final modeling step is specifying *parameters*. As explained earlier, when building a BN only prior (for root nodes) and conditional (for other nodes) probability distributions are needed. Parameters can be a) elicited from human experts, b) learned from available data, or c) a result of a combination of elicitation and learning. For instance, parameters would be learned from data and human experts would subsequently inspect and tune them fine. Keeping a human in the loop is a good idea as the means of avoiding obvious discrepancies between the real world and the model.

We use the sneezing example to illustrate the parameter elicitation process. Since all variables are binary we need to distinguish between only two states they can take on: present and absent. Whenever a name of a variable appear on its own it denotes the state present. We preceded variable names with $\sim$ to denote the complementary state, absent.

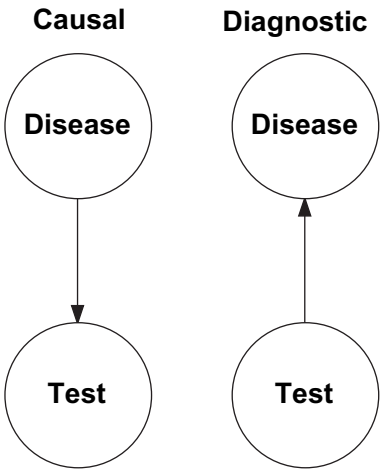

Fig. 3. The two arc direction options.

Prior probability distribution is an unconditional distribution that does not change, no matter the observations we make. Cat is one of the two root nodes, in our example network. We obtain the probability distribution for that node by answering a question "how likely is a couple (like Paul and Susan) to have a cat? " The answer will be a number between 0 and 1 (or between 0 and 100%, which can later be transformed into a probability by dividing it by a 100). Let us take a guess that about 20% of couples own cats. Therefore, we arrive at the probability of 0.2 that Paul and Susan have a cat.

Conditional probability distribution changes with the changes in factors it is conditional on. In the sneezing example, we need to provide the probability of Sneezing given Cold and Rhinitis. Both Cold and Rhinitis can induce sneezing independently of each other. From the point of view of Sneezing, the interaction between them resembles an OR-type influence. That is, the presence of either can cause sneezing to be present, and the presence of both further increases the odds of sneezing. Below are example probabilities capturing this sort of influence.

$P(Sneezing|Cold, Rhinitis) = 0.99$
$P(Sneezing|Cold, \tilde{}Rhinitis) = 0.85$
$P(Sneezing|\tilde{}Cold, Rhinitis) = 0.9$
$P(Sneezing|\tilde{}Cold, \tilde{}Rhinitis) = 0.01.$

Sneezing is very unlikely to happen in the absence of both the factors (0.01). Furthermore, Rhinitis has a slightly higher chance of causing Sneezing than does Cold (0.9 versus 0.85). When both factors are present, Sneezing is bound to occur (0.99).

Two of the probabilities above are quite extreme. Despite that, they still leave some room for uncertainty. For example, John will be Sneezing, on average, in one out of a 100 situations when he has not caught Cold and was not having Rhinitis. Assigning probabilities of 0 or

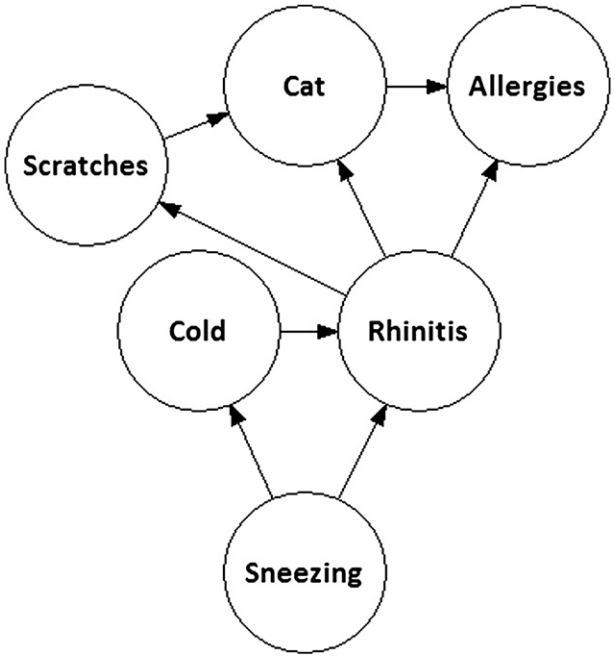

Fig. 4. A Bayesian network equivalent to the one for the sneezing example from Fig. 2(e), but with all arcs reversed.

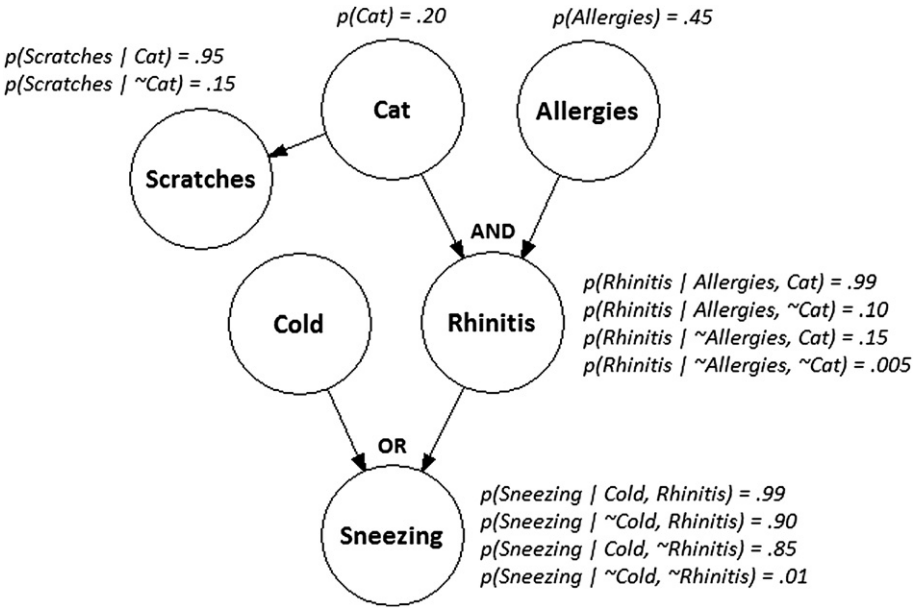

Fig. 5. The sneezing example model with parameters.

1 to 1 of the states of a variable precludes the occurrence of any other state that variable might have (probabilities always sum up to 1). Those values make variables deterministic. Since a model cannot account for all possible causes of any particular event (the purpose of modeling is to simplify and accentuate only the most relevant aspects), it is usually a good idea to avoid making variables deterministic.

A knowledge engineer should always strive to minimize the number of parameters in the model. Those savings become apparent in the case of more complex models. Those models become easier to understand, debug, fine-tune, and solve. OR- and AND-like relationships can be represented in a more compact way using so called *canonical gates*: the noisy-OR and the noisy-AND. While the discussion of those gates is beyond the scope of this manuscript, an interested reader can find more information in Díez and Druzdel (2002).

Fig. 5 shows the sneezing example model with all probability distributions. One notable fact is that the relationship between Cat, Allergies, and Rhinitis is modeled to resemble an AND-like interaction. In our model, Rhinitis can happen only in the presence of a Cat and only in individuals suffering from Allergies.

3.4. Independence relationships

As mentioned earlier in this section, the absence of an arc between X and Y means that X is not a direct cause of Y (and vice versa). However, variables can be connected through other variables. The notion of *conditional independence* is at the center of cases like that.

Fig. 6 shows the possible configurations of three adjacent variables that can be found in a BN. In the *chain* configuration (also called *linear* or *serial*), A and C are dependent if we do not know which state B is in. Once we know the state of B, A and C become independent. The state of C is influenced only by the state of B, and no change in A is carried over to C. We say that C is independent of A given B. In the sneezing example, a chain is formed by Allergies, Rhinitis, and Sneezing. If we do not know whether or not John is having Rhinitis, knowing whether or not he has Allergies influences our belief about him Sneezing. If he has Allergies, probability of Sneezing is higher; if he has not, it is lower. However, once we know that John is not having Rhinitis whether or not he has Allergies becomes irrelevant from the point of view of Sneezing.

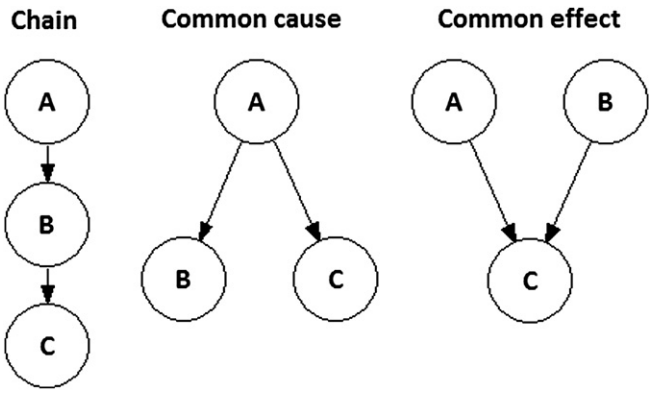

Fig. 6. Possible configurations of three adjacent variables in a Bayesian network.

In the *common cause* configuration (also called *diverging*), *B* and *C* are dependent if we do not know which state *A* is in. Once we know the state of *A*, *B* and *C* become independent. If we don't know if the common cause *A* is in effect or not, observing *B* changes the probability of *C* and vice versa. Once we observe *A*, the effects are not dependent any more. In the sneezing example, Cat is a common cause of Scrathes and Rhinitis. If we do not know if a Cat is in the house, once we notice (observe) Scratches, Rhinitis becomes more likely. Once we get to know that a Cat is there, we have established out belief about Rhinitis (i.e. Rhinitis is more likely). Noticing Scratches after that does not change our belief with respect to Rhinitis, because we already know that a Cat is present.

In the *common effect* configuration (also called *converging*), *A* and *B* are independent in the case of no observation regarding the common effect *C*. Once we know the state of *C*, *A* and *B* become independent. If we know that the common effect is in action, observing one of the causes being in effect will *explain away* other causes. In the sneezing example, Cold and Rhinitis are independent if we do not know if John started Sneezing. However, once he starts Sneezing the fact that he has Cold will drive the likelihood of Rhinitis down. Rhinitis will be explained away by Cold.

Dependence between any two variables (or lack thereof) can be thought of as the ability (or inability) of one of those variables to influence the other. Any two dependent variables respond together to a change in either of them, no matter what the direction of that response be. This influence (or lack of it) can be visualized as a communication channel between those variables. This channel can be open or closed. It is closed between two independent variables and open between dependent ones.

3.5. BN in action: an example

Once a BN has been constructed, it is ready for inference. The inference commences by computing the probability of each state of each variable in the network. After some observations (evidence) are introduced, posterior probabilities are computed. Those probabilities can then be used to determine the most likely causes of a certain event or a set of events (*diagnostic reasoning*) or to predict the results of a test or a set of tests (*predictive reasoning*).

Fig. 7 shows the evolution of the model's parameters as new evidence is introduced. The first step is the computation of all a priori probabilities. This step is commonly called the initialization process, and its results are known as the initial state.

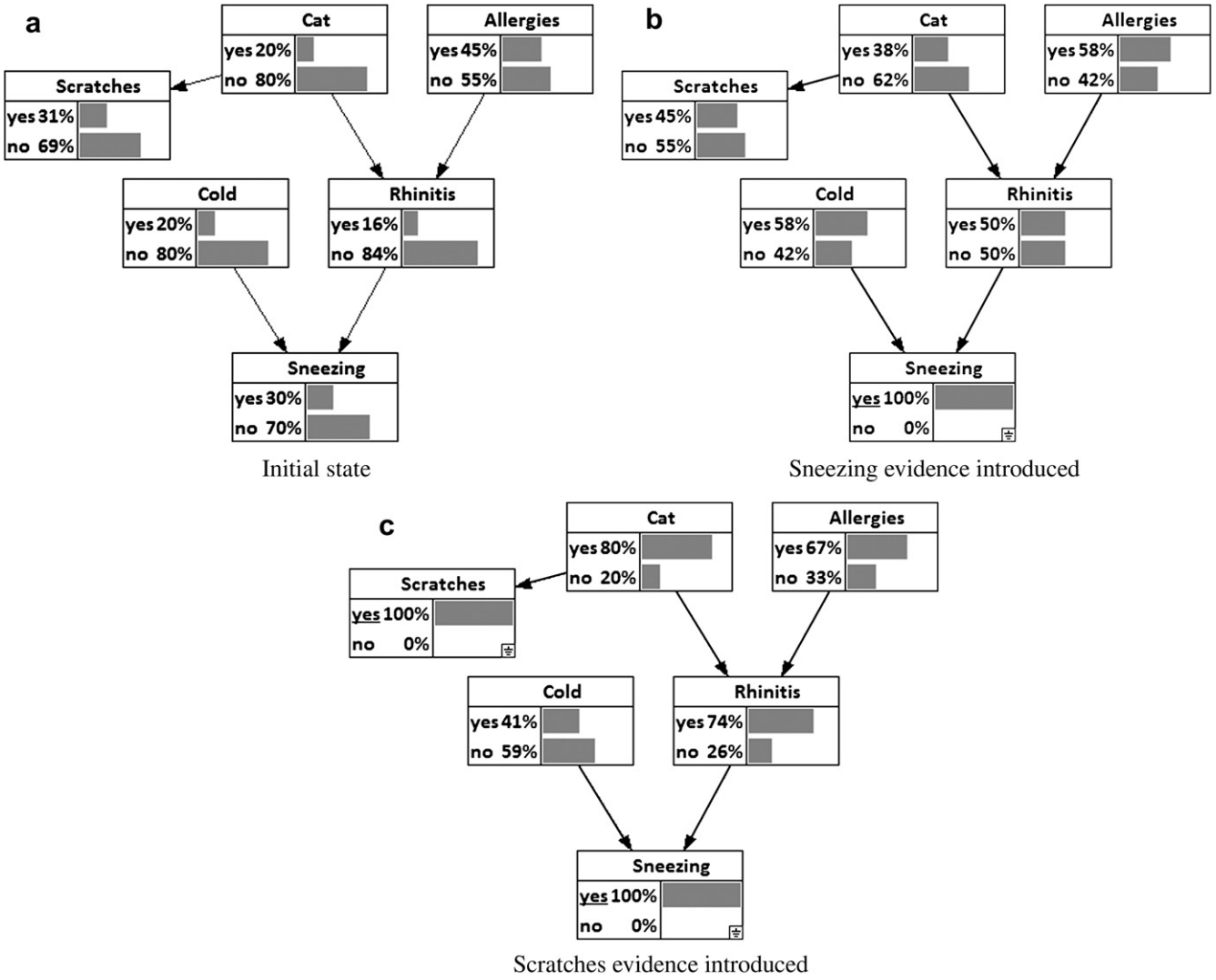

Fig. 7. Probability distributions associated with variables in the sneezing example at different stages of inference.

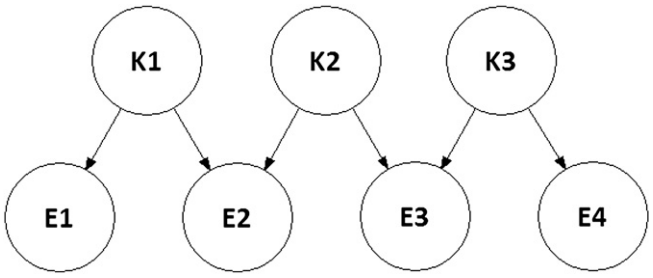

Fig. 8. A simple two-layer student model.

When John starts Sneezing the probabilities are updated to account for this information. Note, that the probabilities of the positive states of all nodes in the network are higher. This is because the evidence available ("John is sneezing") supports the positive state of all the variables.

The next piece of evidence available is "John notices scratches." After this observation is introduced into the model, the probability of a Cat increases, because the evidence of Scratches being present increases the likelihood of a Cat being present. The new and more probable Cat further increases the likelihood of both Rhinitis and Allergies being present. The probability of Cold decreases since it is explained away by Rhinitis (their common effect, Sneezing, has been observed already).

4. Student model building

In this section we shed light on building Bayesian models for student modeling. We use causal arc direction for connecting variables.

4.1. Variables

As explained in Section 3.3.1, variables in a BN can be divided into targets, observations, factors, and auxiliary. Target variables represent features that a system will use to customize the guidance of or assistance to the student. Examples of user features commonly used in student modeling include:

- Knowledge: propositional (the student knows p) or procedural (the student has mastered the rule r).
- Cognitive features: learning styles, cognitive and meta cognitive skills, etc.
- Affective attributes: self-image, motivation, emotional state, etc.

Evidence variables represent all directly observable features of the student's behavior. Examples of those include:

- Answers to questions.
- Measurable traits of conscious behavior: pages visited, time elapsed, hints requested, etc.
- Measurable values of unconscious response: eye movements, physiological data, etc.

Factor variables represent features of the situation the student is or was in that affect other variables, but are of no direct interest otherwise (i.e., they are not direct components of any decision, but affect). Depending on the case at hand, the same variable could be a target or a factor variable.

Another relevant distinction is that between global and local variables (Reye, 1996, 1998, 2004). *Global variables* represent overall features that are linked to many other nodes. For example, psychological states (bored, interested, etc.) or cognitive skills. *Local variables* are linked to a modest amount of target variables; for example, the answer to a test item usually is linked to several (sometimes one) pieces of knowledge.

Finally, a distinction should be made between *static* and *dynamic* variables. A student model is updated with new evidence. Additionally, the student's state usually changes as a result of the interaction with a system. Therefore, some mechanism is needed to take into account the expected changes in the student's state. Dynamic Bayesian networks (Dean & Kanazawa, 1989) are usually employed to address that problem (Reye, 1996, 1998, 2004).

4.2. A simple student model

Once variables of a student model have been defined, the simplest way to connect them is by constructing a two-layered BN depicted in Fig. 8. Target variables (K_1, …, K_n) are parents and evidence variables (E_1, …, E_m) are children of one or more target variables.

When K_1, …, K_n represents pieces of knowledge (propositional or procedural) and E_1, …, E_m are student's responses (e.g., answers to test questions or submissions of problem solutions) we obtain a typical instantiation of this scheme. Knowledge is represented by hidden variables whose state cannot be observed directly and need to be inferred. A link between K_i and E_j means that K_i causes the type of responses coded as E_j. The knowledge modeled as K_i can be wide (e.g., history of the US) or narrow (e.g., the name of the second president of the US).

In the simplest model, all variables are binary: K_is can be known or not-known, while the evidence provided by E_js can be positive or negative. In what follows, we adhere to a quite standard notation of denoting positive states (known, yes, etc.) by 1 and negative states (not-known, no, etc.) by 0.

This simple model can be instantiated in many other ways. For example, K_1, …, K_n could be learning styles or other individual traits, and E_1, …, E_m could be the results of comprehension tests or observations concerning the student's behavior.

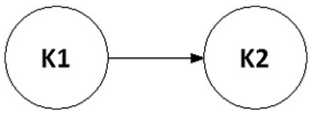

Fig. 9. Prerequisite relationship between two pieces of knowledge.

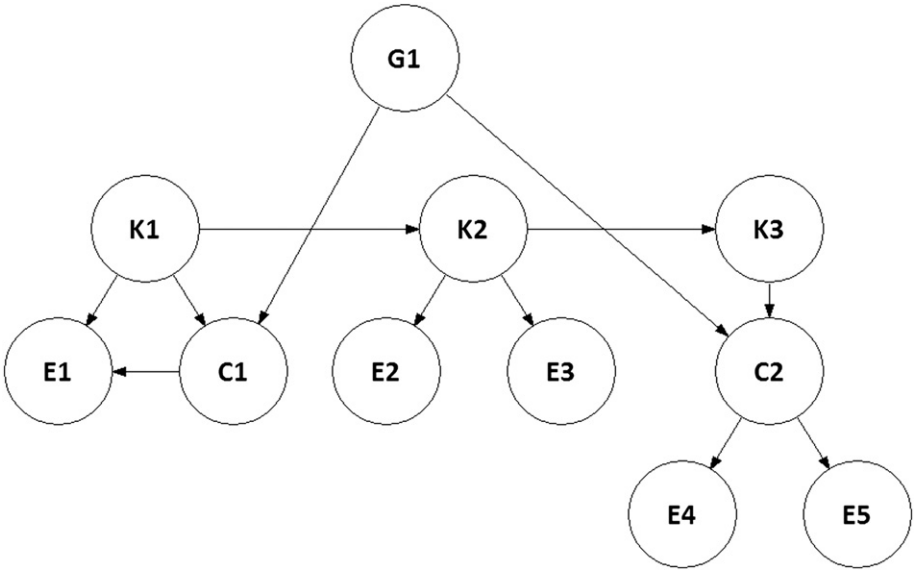

Fig. 10. A Bayesian network backbone for prerequisites.

For evidence variables with multiple parents, the influence of knowledge variables can be of different nature. In the case of the AND-type influence, all K_is have to be known by the student in order for them to provide the correct response E_j. In the case of the OR-type influence, it is sufficient to know at least one K_i to provide the correct response E_j. Those and other influence patterns are modeled by appropriately shaping the parameters (conditional probabilities) of evidence variables.

Sometimes all K_i are modeled as a single variable K. Then K will not be binary. This is the case when K is the scalar "latent trait" known from psychometric literature. This is also the case when the purpose of the model is to classify a student as belonging into a set of stereotypes (or classes). Such a BN is known as a *naive Bayes* classifier (Duda & Hart, 1973).

4.3. Usual links

In this section we explain how to build models with the common types of links encountered in student modeling.

4.3.1. Prerequisite relationships

Prerequisite relationships define the order in which learning material is believed to be mastered. For example, a model could enforce a constraint that it was necessary for the student to be able to add before they could learn to multiply (if they cannot add they cannot multiply). A piece of knowledge K_1 is said to be a prerequisite of a piece of knowledge K_2 if K_2 cannot be acquired prior to K_1.

Prerequisite relationships are very useful in student modeling since they can speed up the inference. For example, if K_1 is a prerequisite of K_2, and the model is certain that the student has not learnt K_1 yet, then the odds are K_2 is not known to them either. This can be modeled using conditional probabilities. The probability of $K_2 = 1$ given that $K_1 = 0$ should be assigned a value close to 0, for instance, $p(K_2 = 1|K_1 = 0) = 0.05$. Then, the probability of knowing K_2 when K_1 is known would be given as $p(K_2 = 1|K_1 = 0.95)$.

When modeling prerequisite relationships using the causal arc direction we state that knowing K_1 influences knowing K_2 (Fig. 9). It could be argued that the concept of cause is not entirely suitable here. However, according to Reye (2004), "(…) when causality is seen not just to involve factors that have a positive influence, buy also factors that have a negative influence, then not knowing topic A is a cause for not knowing topic B."

Reye (Reye, 1996) proposed the concept of "belief net backbone[3]. In that network, all knowledge nodes K_i are connected by a set of arcs which represent the prerequisite relationship. A "topic cluster", composed by pieces of evidence E_j and intermediary variables C_k, is connected to every knowledge variable K_i. A topic cluster is a small network of evidence and intermediary variables related to K_i. All clusters are pairwise disjoint. Finally, it is possible the presence of a few global nodes G_i whose state affects many other nodes (Fig. 10).

Knowledge variables (Fig. 10) form the backbone of the network. As claimed by Reye, this backbone approach has several advantages. First, it gives the student model designer a standard structure for the network. Second, there are computational advantages due to the fact that the updates of the beliefs in any topic cluster only affect the other topics via the backbone. However, it also has some limitations. The

[3] Note, that Bayesian networks have been referred to as Bayesian belief networks Pearl (1986) or simply belief networks Cooper (1990), among other names.

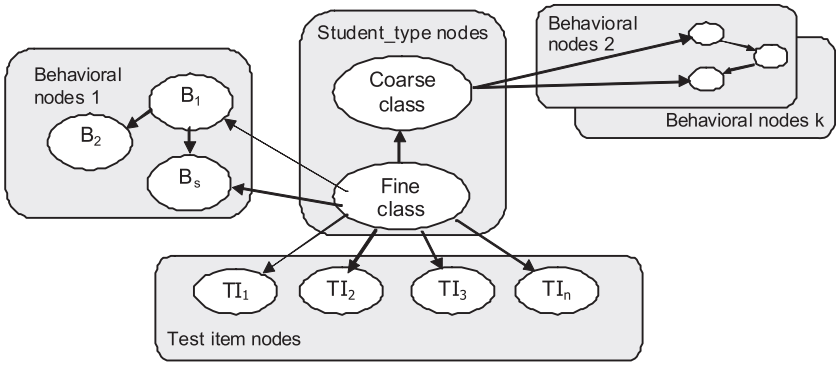

Fig. 11. A Bayesian student model based on misconceptions.

biggest one is its limited expressive power. The model does not allow for representation of more complex dependencies between the variables, e.g., evidential variables depending on more than one knowledge node.

4.3.2. Refinement relationships

Another relationship ubiquitous in student modeling context is that of *refinement*. Student knowledge or individual traits can be described at different levels of detail. A piece of evidence could help to classify the student at the finest level of detail, while a different piece of evidence could discriminate only at the coarsest level of detail.

A good example of this is the student model for DCT, a tutoring system on decimal numbers (Stacey, Sonenberg, Nicholson, Boneh, & Steinle, 2003). In that system, there are several types of variables (Fig. 11; after Stacey et al., 2003). Evidential variables can be either test items of a decimal comparison test (denoted as T_i) or can account for the student's behavior during their interaction with the system (denoted as B_j). The model also contains student-type variables. The students are classified according to their misconceptions (this model belongs to the category of bug models). For example, class *L* groups students that think longer decimals are larger numbers, e.g., 3.145 is greater than 3.9. This classification happens at two levels of granularity: coarse (4 classes: *L*, *S*, *A*, *U*) and fine. For example, the mentioned coarse class *L* has the fine subclasses *LWH*, *LZE*, *LRV* and *LU*. Each of these classes represents a particular misconception related to coarse class *L*. There is a link from the node representing the fine class to the variable representing the coarse class.

The system contains several groups of behavioral variables. Those groups can be of different sizes and can have different internal structures (i.e., they can be connected in different ways). They model different instructional episodes that can help to classify the student's misconceptions. Please note, that there is only one group of test item and student-type variables. The performance of this student model has been evaluated by comparing the results of manual expert diagnosis with diagnosis performed by the system. The comparison took into account the results of more than 2000 students who took the test administered by the DCT system and showed an 80–90% agreement rate.

4.3.3. Granularity relationships

Granularity relationships describe how the domain is decomposed into components. The decomposition can be *coarse-grained* if it consists of fewer, larger components, or *fine-grained* if there are multiple levels of decomposition leading to more components of smaller size (Greer & McCalla, 1989). When developing a student model, it is important to make a decision about the level of granularity needed early in the process.

If a piece of knowledge *K* is divided into several pieces of knowledge $K_1, \ldots, K_n$, then using the causal arc direction implies, that knowing the smaller pieces K_i causes the student to know *K* (Fig. 12). For further discussion about the adequate direction of the links for this case, (see, for example, Collins, Greer, & Huang, 1996; Millán and Pérez-de-la Cruz, 2002).

A problem can arise when one attempts to simultaneously model prerequisite and granularity relationships. If both are included in the same model, links with different interpretation are mixed and the model gets difficult to build and understand. For example, let us assume we have a compound concept *K* which is composed of two sub-concepts, K_1 and K_2. We also have a piece of knowledge *P* that is a prerequisite of *K*. Fig. 13 shows a BN for this example.

The problem with that network is that the conditional probability of *K* given its parents is difficult to specify. A possible solution is to group variables of the same type by introducing intermediate variables. We could create a concept K_{12} that models K_1 and K_2 jointly, as shown in Fig. 14. The new model implies, that *K* will be known once both K_{12} and the prerequisite *P* are known.

Sometimes, a piece of knowledge *K* can be acquired in one of many ways. Fig. 15 shows an example of a model with two alternative ways. The first way of acquiring *K* is through learning of the complete set of prerequisites $P_1, \ldots, P_n$, modeled jointly as *P*, and a compound concept

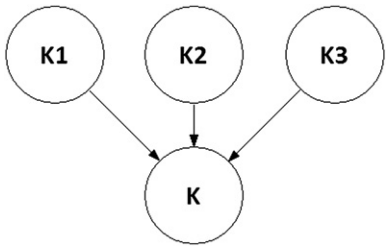

Fig. 12. A Bayesian network modeling granularity relationships.

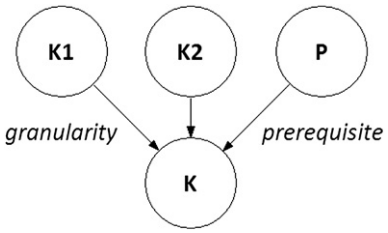

Fig. 13. A Bayesian network modeling granularity and prerequisite relationships simultaneously.

K_{12}, that groups two sub-concepts K_1 and K_2. The second way is through mastering the same set of prerequisites $P_1, \ldots, P_n$ (P) and a certain concept C.

4.3.4. Time factor

Dynamic models are a class of a potent alternative for modeling of relationships between knowledge and evidential variables. Those models allow for changes in the state of variables over time. Such modeling of the evolution of the student's state matches the dynamic nature of the learning process. In dynamic graphical models known as *dynamic Bayesian networks* (DBNs; Russell & Norvig, 2002) time is discrete and a separate BN is constructed for each step (or time slice). This approach has been taken by several authors (e.g., Manske & Conati, 2005; Mayo & Mitrovic, 2001; Millán and Pérez-de-la Cruz, 2002; Zapata-Rivera, 2004), likely inspired by the work of Reye (1998, 2004). Reye presents a simple model based on DBN which captures the dynamic nature of the student's knowledge (Fig. 16). In that proposal, for each $j = 1, \ldots, n, \ldots$, two types of variables are defined: (1) L_j is the student's state of knowledge after the j-th instance of an interaction with the system, and (2) O_j is the result of the j-th interaction.

In Reye's model, at the time step n, the probability that the student is now in the learned state (L_n) depends on whether the student was already in the learned state at the previous time step (L_{n-1}) and on the outcome of the last interaction with the system (O_n). Reye provides formulas for a two-phase updating of this model. Please note, that in spite of the fact that this description seems to refer to only one knowledge variable, the implementations of this model will contain as many knowledge nodes as needed, resulting in potentially complex networks.

A study reported in Millán, Pérez-de-la Cruz, and García (2003) attempted to empirically compare a static BN with a DBN using simulated students. The results obtained show, that both approaches can produce a very similar performance in terms of test length and accuracy, although the static model seems to perform slightly better.

4.4. Learning from data

BNs described in earlier sections present perhaps the most typical examples of the relationships between variables in a student model. However, in general, any two variables could be connected by an arc. Therefore, a problem arises of determining which links should be considered and which should be removed.

As mentioned in Section 3.3, machine learning techniques can be used to infer the structure and/or parameters of a BN. Machine learning techniques take a data set as an input. That data set is a collection of configurations of states of variables of interests as registered in the real world. For instance, if we were interested in modeling the influence of emotional state on learning progress, we could have several students attempt to learn a given piece of learning material. We would acquire a sample of their emotional state every time they performs an observable action, e.g., provide an answer to a test item.

Results obtained from applying fully automatic methods can be difficult to understand and justify by humans. Learning of both the structure and parameters of a BN is usually performed only in cases of the absence of prior knowledge about the problem being modeled or in cases of very large and complex problems. In most of other cases, a knowledge engineer controls the learning process.

One way of making the machine learning process more controlled is to introduce structural constraints (enforce or prohibit links between certain variables) and learn the reminder of the structure along with parameters or learn exclusively the parameters. The accuracy of the resulting model can be subsequently investigated.

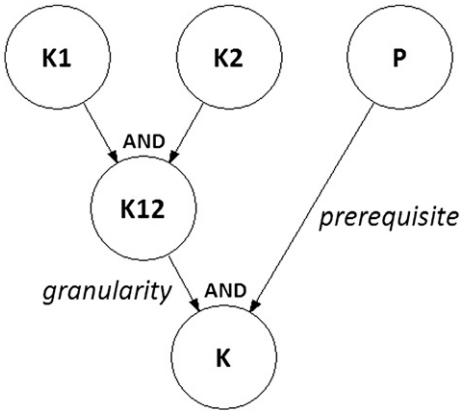

Fig. 14. A Bayesian network modeling granularity and prerequisite relationships simultaneously – with intermediate variable introduced.

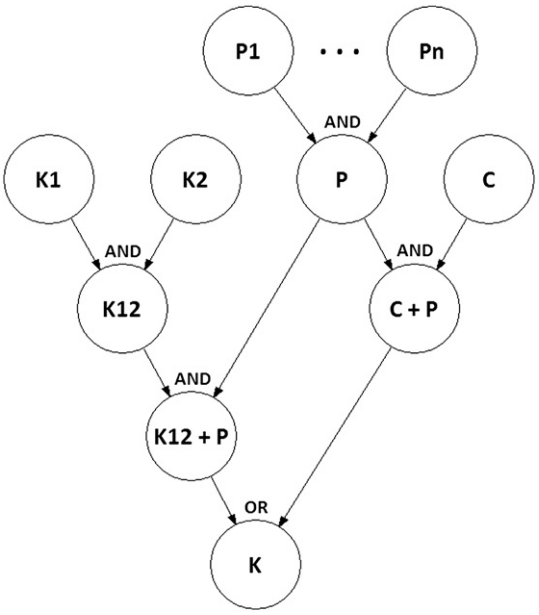

Fig. 15. A Bayesian network modeling two ways of a learner's knowledge acquisition.

Structural constraints can be a result of applying prior knowledge (e.g., coming from prior studies). Alternatively, it can be a result of an exploratory analysis of the collected data set performed by a knowledge engineer. Such an analysis may aid in gaining insight into the possible relationships between variables. We will illustrate this methodology using a BN to infer student's learning attitudes from a log file described in Arroyo and Woolf (2005). The network was created using log data from Wayang Outpost, a multimedia web-based tutoring system for high school mathematics. The authors considered *observation nodes* such as average number of incorrect responses, average number of hints in assisted problems, etc. They also considered a set of 12 *hidden target variables* to model the student perceptions and attitudes. The values of these variables were obtained from each student through a post-test survey.

A preliminary analysis was performed in order to identify statistical correlations between all pairs of variables. This analysis produced a set of potential links. Some of these links were then selected in order to define a DAG, according to the following procedure:

1. Eliminate links between observable variables.
2. Set the causal direction for links between hidden and observable variables.
3. Select the more intuitive causal direction for every correlation between hidden variables.
4. Eliminate cycles by removing the weakest links.

Once the structure is determined in this way, machine learning techniques can be applied to log data in order to estimate numerical parameters.

Finally, there are several works that attempt to use log data to evaluate the appropriateness of a Bayesian learner model. In Ting and Phon-Amnuaisuk (2009), log data was pre-processed, transformed and fed into three different candidate DDN models. The performance of the three candidate models was analyzed and compared, in order to obtain the optimal Bayesian learner model. And in Yudelson, Medvedeva, and Crowley (2008), a multifactor approach is used to evaluate models with different modifications of the classical Bayesian Knowledge Tracing model (Corbett et al., 1995) in order to select an appropriate student model for a medical ITS.

4.5. More complex models

So far, we have discussed several simple BNs used to represent student's features (mainly knowledge). The goal of this section is to present a selection of more complex BNs that have been used to model a wide range of features and processes, such as: problem solving, metacognitive skills, and emotional state and affect.

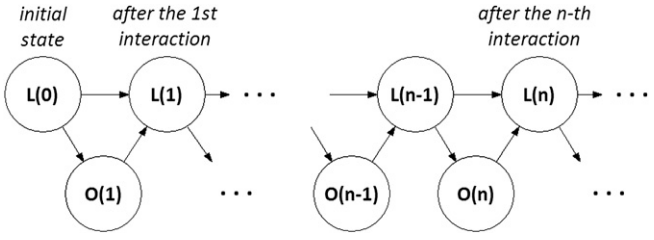

Fig. 16. A dynamic Bayesian network for student modeling.

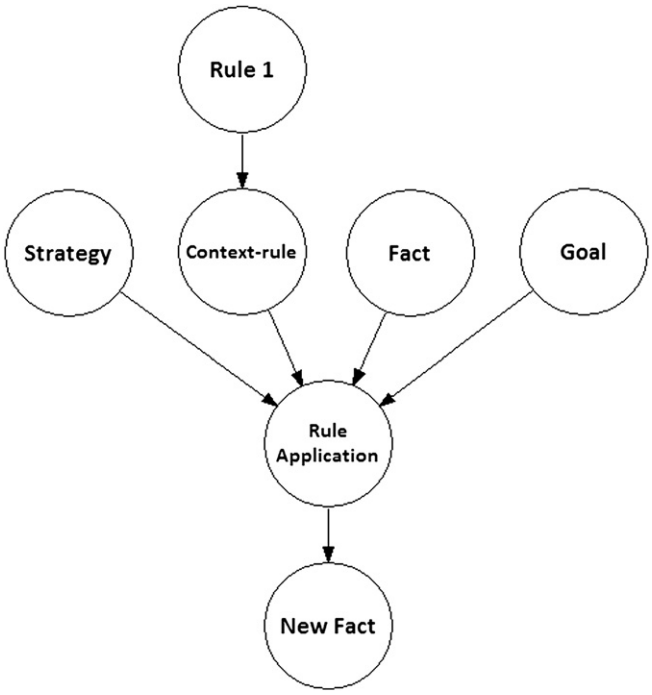

Fig. 17. Basic structure of ANDES BNs.

4.5.1. Problem solving

A good example of modeling the problem solving process is the physics tutor ANDES (Conati et al.1997, 2002). The student model of ANDES allows three kinds of assessment: a) plan recognition (discovering the most likely problem solving strategy that the student is employing at a given point in time), b) prediction of student goals and actions, and c) long-time assessment of the student's knowledge. A BN implemented in ANDES includes variables of the following types:

- *Knowledge variables*. The knowledge in the domain of physics has been modeled using *rules* (e.g., $F = m \times a$), *facts* (e.g., $m = 50$ kg and $a = 4$ m/s^2), and *context-rule* nodes (e.g., the application of the rule $F = m \times a$ in a given context).
- *Goal variables* that represent the goal the student is pursuing (e.g., computing the force F).
- *Strategy variables* that denote the student's choices among alternative plans to solve the problem at hand.
- *Rule application variables* that model a derivation of a new rule given the strategy, goal, rule, and facts.

ANDES builds a separate BN for each physics problem approached by the student. Each of those BNs has up to two hundred nodes, depending on the complexity of the problem. The basic structure of such a network is shown in Fig. 17.

ANDES assumes, that if the student knows a certain rule (e.g., $F = m \times a$) they will be able to use it in any context. Then, if they also know a certain set of facts (e.g., $m = 50$ kg and $a = 4$ m/s^2), have a certain goal (e.g., compute the force F), and employ a certain strategy (way of approaching and solving the problem), then the rule will be applied to the problem and a new fact will be inferred (e.g., $F = 200N$).

An approach similar to the one implemented in ANDES has been used more recently to model problems in the domain of medical diagnosis (Suebnukarn & Haddawy, 2006). Instead of rules, however, a BN includes "medical concepts" that can be applied to derive a hypothesis (equivalent to facts in the domain of physics). *Application* and *Goal* variables are used in the same way as in the ANDES system. The medical diagnosis model differs from the ANDES model in two important ways: (1) *Strategy* variables are not included and (2) the causal relationships between hypotheses are modeled.

The use of such complex models allows a diagnosis of both the student's knowledge and their problem solving procedure. In this way, a system can provide personalized assistance and coach the student aiding them in reaching a goal they set out to achieve.

4.5.2. Metacognitive skills

Metacognition refers to the awareness of one's ability to understand, control, and intervene actively in their own learning process. A person with good metacognitive skills is a person that has learned how to learn. Today, learning cumulated in the earlier stages of life is very often not enough, so improving metacognitive skills has become very important and as such an object of increased interest of many researchers.

In the field of computer based education, some Bayesian student models have been developed to support the application of metacognitive skills. A good example is the support for the Analogical Problem Solving (APS). In cite Muldner and Conati (2005), APS was implemented as a computational framework designed to provide adaptive support for using examples during problem solving activities. Users of the system are classified into two groups depending on the strategy they use. *Min-analogy* identifies students that try to solve the problem on their own and consult the solved examples only after reaching an impasse. *Max-analogy* students copy as much as possible instead of using their own capabilities. Min-analogy is a more effective learning strategy, as it provides opportunity to reinforce knowledge during practice and bridge knowledge gaps.

A good practice during reasoning about an example solution is Explanation Based Learning of Correctness (EBLC), which is a form of self-explanation used to overcome learning impasses when knowledge is insufficient to understand a solved example. To model EBLC and APS, new variables have been added to the ANDES system:

- *Copy variables* that encode the fact of the student copying a step. Those variables have the following states: correct, incorrect, and no-copy.
- *Similarity variables* that measure the similarity between the solved problem fact and the corresponding fact in students solution. Those variables have the following states: trivial, non-trivial, and none.
- *Analogy-tend variables* that represents the student's preferred APS style. Those variables have the following states: min-analog and max-analog.
- *EBLC variables*, a binary variables that indicating whether the student has explained the step through EBLC.
- *EBLC-tend variables* that measure the student's EBLC tendency.

Fig. 18 shows the structure of the part of that BN that supports the evaluation of the APS metacognitive skill. This part is connected to the student's knowledge BN by links from copy variables to fact variables. The BN for APS allows the system to support tailored interaction with the user by means of hints and the retrieval of appropriate examples.

4.5.3. Emotions

Recently, there has been increasing interest in widening the range of user's characteristics accounted for by computer applications (Conati & Maclaren, 2009; Muldner, Christopherson, Atkinson, & Burleson, 2009, chap. 15, pp. 138–149). This trend is very visible in modeling features that can have an impact on human performance. Emotional states and affects have accumulated good amount of attention.

A good example is the affective user model in Prime Climb, an educational game designed to help children learn number factorization while being coached by an intelligent agent (Conati & Maclaren, 2009). The Bayesian student model of Prime Climb includes several types of variables:

- *Goal variables* that model the player's objective (e.g., learn math, have fun, beat the partner).
- *Action variables* for both the player and the action.
- *Goal satisfaction variables* to encode the degree of satisfaction that each action causes.
- *Emotional variables* that represent six of the 22 emotions described in the Ortony, Clore, and Collins's (OCC) theory of emotions (Ortony, Clore, & Collins, 1988).

The following emotions are modeled in Prime Climb:

- Joy/distress (states of the node Emotion for the game).
- Pride/shame (states of the node Emotion for self).
- Admiration/reproach (states of the variable Emotion for agent).

The structure of the BN is depicted in Fig. 19. The system uses the student model to boost their learning progress while maintaining high levels of engagement and satisfaction achieved by the game play.

5. An example: developing a BSM for linear programming

In this section we will try to illustrate the proposed student model building process by applying it to define a BSM for the Simplex algorithm for Linear Programming problems.

We have chosen the Simplex algorithm[4] domain for two reasons: a) there are previous efforts in developing tutors for such domain (López, Millán, Pérez de la Cruz, & Triguero, 1998; Millán, García-Hervás, Guzmán De los Riscos, Rueda, & Pérez de la Cruz, 2004); and b) the domain is procedural rather than declarative (building a model for a procedural domain seems a more challenging task).

In procedural domains, the necessary knowledge to solve a linear programming problem is better described as skills. Each one of these skills corresponds to a necessary step in the algorithm. The steps need to be performed in a certain order to achieve the optimal solution of the problem presented.

The graph in Fig. 20 shows a possible learning strategy for learning the simplex algorithm[5]. This learning strategy was created by an Operations Research instructor.

In this learning strategy we can see that the problem-solving procedure has been decomposed in basic skills (round nodes) and types of problems (square and shaded nodes). Fig. 20 shows which skills are necessary for solving each of the eight different types of problems. For example, square node 1 shows that to be able to solve a maximization linear programming problem with unique solution, it is necessary to have skills 1–7. The levels are cumulative in the sense that each level of problems requires also to be able to solve problems of all the preceding levels. So, for example, to be able to solve problems in node 6, it is necessary to have all skills required for problems 1 to 5, and also skills 11, 12 and 13.

This learning strategy was implemented in López et al. (1998) and Millán et al. (2004), but it was only used to select the next instructional action (which material should be presented or which problem should be posed next). The student was presented with a problem belonging to the simplest class of problems. Once the student showed his/her competency by solving several problems of this type, he would be taught

[4] The Simplex algorithm was defined by Dantzig in 1944. It was selected in year 2000 as one of the Top Ten algorithms in the 20th century Cipra (2000).

[5] LP_0 is a linear programming maximization problem, with non-negative variables and resources, and with $\leq$ constraints.

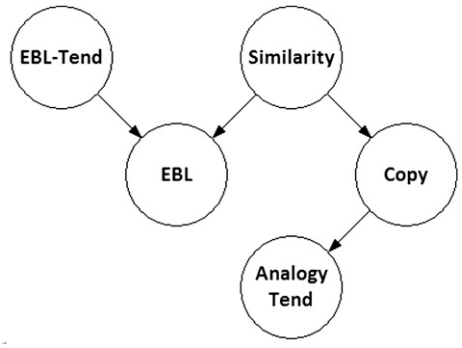

Fig. 18. A BN supporting the Explanation Based Learning of Correctness (EBLC).

the next set of skills and then posed a problem involving all the learned skills. The procedure continued until the student masters all the eight skills.

To implement this learning strategy by means of a Bayesian Student Model, we recall that, as explained in Section 4.2, nodes can represent a different range of knowledge variables, ranging from concepts to skills, abilities, etc. So in this case we can define several nodes for our BN:

- Nodes to represent skills. For example, for each $i = 1, \ldots, 15$ we define the propositional variables S_i as $S_i = 1$ if the student has skill number i, otherwise $S_i = 0$.
- Nodes to represent abilities. Such nodes will account for the student having an ability (which in this context we define to be a certain set of skills) necessary for solving certain kind of problems. In this way, we can define seven abilities:

By the moment, the nodes defined account for the skills and precedence relationships to be taken into account when teaching the simplex algorithm. More nodes are needed to collect evidence about a student:

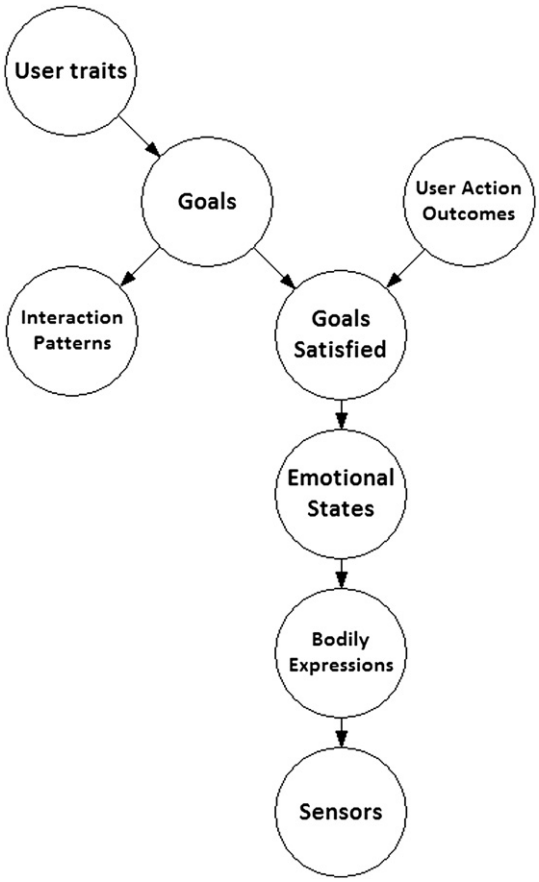

Fig. 19. A Bayesian network for the Prime Climb game.

- Evidential problem nodes. As the simplex algorithm is a procedural domain, the evidential nodes are problems to be solved by the student. We therefore define propositional variables:

$A_1 = 1$ if the student has all skills 1–7	0 otherwise
$A_2 = 1$ if the student has ability A_1 and skill 8	0 otherwise
$A_3 = 1$ if the student has ability A_1 and skill 9	0 otherwise
$A_4 = 1$ if the student has abilities A_2 and A_3	0 otherwise
$A_5 = 1$ if the student has ability A_4 and skill 10	0 otherwise
$A_6 = 1$ if the student has ability A_5 and skills 11, 12, 13	0 otherwise
$A_7 = 1$ if the student has ability A_6 and skill 14	0 otherwise
$A_8 = 1$ if the student has ability A_7 and skill 15	0 otherwise

$P_i = 1$ if the student solved problem i correctly, 0 otherwise.

Remember that the ability nodes have been defined as the set of skills necessary to solve certain kind of problems: in this way, there is a causal relationship between the ability and the problem. That is, only students who have the ability (i.e., the necessary set of skills) will be able to solve a particular problem.

- Also, dedicated questions or problems can be used to test a particular skill, in order to get a more precise diagnosis. We will define such questions as:
  $Q_i = 1$ if the student solved problem i correctly, 0 otherwise.

As for parameters, they are easy to specify, because:

- Regarding the relationships between the skills and the ability, in this case they are all of the AND Type (see Section 3.3.4). If desired, a little noise can be introduced in the model to account for lucky guesses and unintentional slips (Baker et al., 2008; Millán & Pérez-de-la Cruz, 2002; VanLehn, Niu, Siler, & Gertner, 1998).
- Regarding the relationship between the ability and the problem, we have that for each $j = 1, \ldots, 8$, $P(P_j = 1/A_j = 1) = 1-\epsilon_j$, and $P(P_j = 1/A_j = 0) = \delta_j$, where ϵ_j, δ_j are typically small numbers (used to model lucky guesses and unintentional slips).
- Regarding the relationship between the skills and the questions, we have that for each $j = 1, \ldots, 15$, $P(Q_j = 1/S_j = 1) = 1 - \epsilon_j$, and $P(P_j = 1/A_j = 0) = \delta_j$, where again ϵ_j, δ_j are typically small numbers.

As usually more than one problem of each type is to be used, in this case it is advisable the use of the rolling-up process (Russell & Norvig, 2002). In the rolling-up process, once the evidence from the problem solving process has been gathered and processed to update the abilities and skills, the evidential node is available again so another problem of the same type can be asked to the student. This allows to keep on gathering useful evidence about his/her state of knowledge. Therefore in this case the use of a dynamic model (Section 4.3.4) is recommended.

Finally the Bayesian network that accounts for a student model for the Simplex algorithm is shown in Fig. 21. So now our BSM for the Simplex algorithm is finished. Recall that in this model there are only specific questions to test skills S_8 to S_{11}.

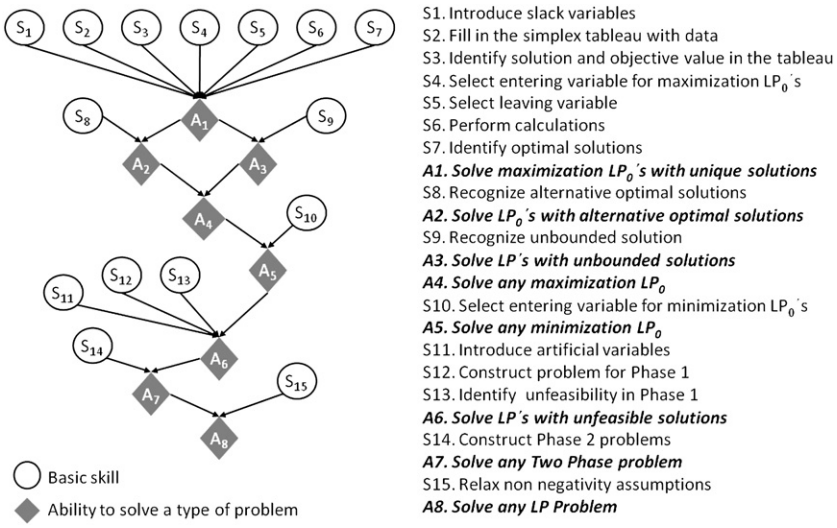

Fig. 20. A learning strategy for the simplex algorithm.

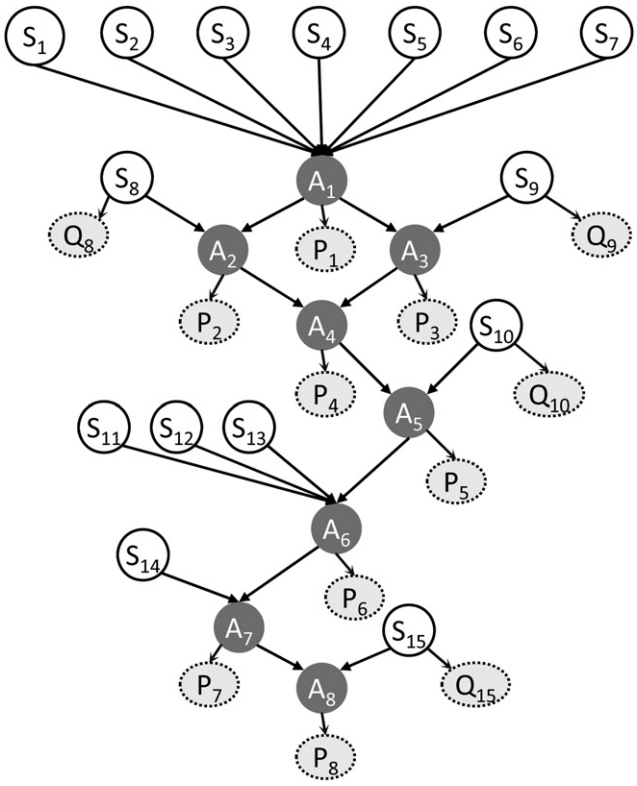

Fig. 21. A Bayesian student model for the Simplex algorithm.

6. Conclusions

User models have become a common component of many computer systems. In some fields such components are not only sought after but rather required. One such field is the field of education.

On the other side, Bayesian networks have proven to be a powerful, sound and versatile technique for any kind of problem involving modeling knowledge. Probably the best evidence of their practical advantages is the great number of successful applications in the last twenty years: they have been widely used with diverse purposes (troubleshooting, diagnosis, prediction, classification) and in very different domains (medical diagnosis, information retrieval, bioinformatics, marketing, etc.) (Pourret, Naïm, & Marcot, 2008).

In this paper, we have introduced concepts and techniques relevant to Bayesian networks, and argued that they are a very suitable tool for forging powerful student models. We have shown examples of networks for typical student modeling situation. We have argued that Bayesian networks can represent a wide range of students' features.

In our opinion, Bayesian networks are expected to play an increasingly important role in the field of student modeling and this paper has attempted to bring them a little closer to education practitioners. Anyone interested in using Bayesian networks can choose among many tools, such as GeNIe and SMILE. Those tools allow for easy model creation and grant access to efficient inference and learning algorithms.

References

Anderson, J., Corbett, A., Koedinger, K., & Pelletier, R. (1995). Cognitive tutors: lessons learned. *The Journal of the Learning Sciences, 4*(2), 167–207.

Arroyo, I., & Woolf, B. P. (2005). Inferring learning and attitudes from a Bayesian network of log file data. *Frontiers in Artificial Intelligence and Applications, 125*.

Baker, R., Corbett, A. T., & Aleven, V. (2008). More accurate student modeling through contextual estimation of slip and guess probabilities in Bayesian knowledge tracing. *Lecture Notes in Computer Science, 5091*, 406–415.

Beck, J.E., 2007. Difficulties in inferring student knowledge from observations (and why you should care). In: Proceedings of AIED2007 Workshop on Educational Data Mining (EDM'07). pp. 21–30.

Brown, J. S., & Burton, R. R. (December 1978). Diagnostic models for procedural bugs in basic mathematics skills. *Cognitive Science, 2*, 155–192.

Brusilovsky, P., & Millán, E. (2007). User models for adaptive hypermedia and? adaptive? educational systems. In P. Brusilovsky, A. Kobsa, & W. Nejdl (Eds.), *The Adaptive Web. vol. 4321 of lecture notes in computer science* (pp. 3–53). Berlin, Heidelberg: Springer.

Burton, R. R., & Brown, J. S. (1982). An investigation of computer coaching for informal learning activities. In D. Sleeman, & J. S. Brown (Eds.), *Proceedings of international conference on Intelligent Tutoring Systems (ITS'82)* (pp. 79–98).

Carbonell, J. R. (1970). AI in CAI: an artificial intelligence approach to computer-assisted instruction. *IEEE Transactions on Man-Machine Systems, 11*(4), 190–202.

Charniak, E. (1991). Bayesian networks without tears. *AI Magazine, 12*(4), 50–63.

Cipra, B. (May 2000). The best of the 20th century: editors name top 10 algorithms. *SIAM News, 33*(4).

Collins, J., Greer, J., Huang, S. (1996). Adaptive assessment using granularity hierarchies and bayesian nets. In: Lecture Notes in Computer Science. Vol. 1086. pp. 569–577.

Conati, C., Gertner, A., & VanLehn, K. (November 2002). Using bayesian networks to manage uncertainty in student modeling. *User Modeling and User-Adapted Interaction, 12*(4), 371–417.

Conati, C., Gertner, A. S., VanLehn, Druzdzel, M. (1997). On-line student modeling for coached problem solving using bayesian networks. In: Proceedings of UM'97, Sixth International Conference on User Modeling. pp. 231–242.

Conati, C., & Maclaren, H. (January 2009). Empirically building and evaluating a probabilistic model of user affect. *User Modeling and User-Adapted Interaction, 19*(3), 267–303.

Cooper, G. F. (March 1990). The computational complexity of probabilistic inference using bayesian belief networks. *Artificial Intelligence, 42*(2–3), 393–405.

Corbett, A., & Anderson, J. (1992). Student modeling and mastery learning in a computer-based programming tutor. *Lecture Notes in Computer Science, Vol. 608*, 413–420.
Corbett, A. T., Anderson, J. R., & O'Brien, A. T. (1995). Student modeling in the act programming tutor. In S. F. Nichols, R. L. Chipman, & R. L. Brennan (Eds.), *Cognitively diagnostic assessment* (pp. 19–41). Erlbaum.
Dean, T., & Kanazawa, K. (1989). *A model for reasoning about persistence and causation*. Tech. rep. Providence, RI, USA: Brown University.
Díez, F.-J., & Druzdel, M. (2002). *Canonical probabilistic models for knowledge engineering*. Tech. Rep. IA-02–01. Dept. Inteligencia Artificial.
Druzdzel, M., Henrion, M. (1990). Using scenarios to explain probabilistic inference. In: Working notes of the AAAI Workshop on Explanation. pp. 133–141.
Druzdzel, M. J., & van Leijen, H. (2001). Causal reversibility in Bayesian networks. *Journal of Experimental and Theoretical Artificial Intelligence, 13*(1), 45–62.
Duda, R. O., & Hart, P. E. (1973). *Pattern classification and scene analysis*. New York: Wiley-Interscience Publication.
Greer, J., & McCalla, G. (1989). A computational framework for granularity and its application to educational diagnosis. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI'89)* (pp. 477–482). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.
Greer, J. E., & McCalla, G. I. (Eds.). (1994). *Student modelling: The key to individualized knowledge-based instruction. Vol. 125 of Nato ASI Series F: Computer and systems sciences*. Springer Verlag.
Jameson, A. (1996). Numerical uncertainty management in user and student modeling: an overview of systems and issues. *User Modeling and User-Adapted Interaction, 5*(3–4), 193–251.
Kahneman, D., Slovic, P., & Tversky, A. (1982). *Judgment under uncertainty: Heuristics and biases*. New York: Cambridge University Press.
Kasurinen, J., & Nikula, U. (2009). Estimating programming knowledge with Bayesian knowledge tracing. *Computer Science Education* 313–317.
Lauritzen, S. L., & Spiegelhalter, D. J. (1988). Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society B, 50*, 157–224.
Liu, C.-l. (2008). Using bayesian networks for student modeling. In R. M. Viccari, P. Augustin-Jaques, & R. Verdin (Eds.), *Agent-based tutoring systems by cognitive and affective modeling* (pp. 97–113). Igi Global.
López, J.-M., Millán, E., Pérez de la Cruz, J.-L., Triguero, F. (1998). Ilesa: a web-based intelligent learning environment for the simplex algorithm. In: Proceedings of International Conference on Computer-Aided Learning and Instruction (CALISCE'98). pp. 399–406.
Manske, M., Conati, C. (2005). Modelling learning in an educational game. In: Proceedings of AIED'05: World Conference of Artificial Intelligence and Education. pp. 411–419.
Mayo, M., & Mitrovic, A. (2001). Optimising its behavior with bayesian networks and decision theory. *International Journal of Artificial Intelligence in Education, 12*, 124–153.
Millán, E., García-Hervás, E., Guzmán De los Riscos, E., Rueda, A., & Pérez de la Cruz, J.-L. (2004). Tapli: an adaptive web-based learning environment for linear programming. In *Current topics in artificial intelligence. Vol. 3040 of lecture notes in artificial intelligence* (pp. 676–685). Springer.
Millán, E., & Pérez-de-la Cruz, J.-L. (June 2002). A bayesian diagnostic algorithm for student modeling and its evaluation. *User Modeling and User-Adapted Interaction, 12*(2), 281–330.
Millán, E., Pérez-de-la Cruz, J.-L., & García, F. (2003). Dynamic versus static student models based on bayesian networks: an empirical study. In *Proceedings of Knowledge-based Intelligent Information and Engineering Systems (KES'03). Vol. 2774 of lecture notes in computer science* (pp. 1337–1344). Springer.
Mitrovic, A. (2003). An intelligent sql tutor on the web. *International Journal of Artificial Intelligence in Education, 13*(2–4), 173–197.
Mitrovic, A., Koedinger, K. R., & Martin, B. (2003). *A Comparative analysis of cognitive tutoring and constraint-based modeling, Vol. 2702*. Berlin, Heidelberg: Springer Berlin Heidelberg.
Muldner, K., Christopherson, R., Atkinson, R., & Burleson, W. (2009). *Investigating the utility of eye-tracking information on affect and reasoning for user modeling, Vol. 5535*. Berlin, Heidelberg: Springer Berlin Heidelberg.
Muldner, K., & Conati, C. (2005). Using similarity to infer meta-cognitive behaviors during analogical problem solving. In: Proceedings of International Conference on User Modeling (UM'05). In *Lecture Notes in Computer Science, Vol. 3538* (pp. 134–143).
Neapolitan, R. (1990). *Probabilistic reasoning in expert systems: Theory and algorithms*. New York: John Wiley & Sons.
Ohlsson, S. (1992). Constraint-based student modeling. *Journal of Artificial Intelligence in Education, 3*(4), 429–447.
Ortony, A., Clore, G. L., & Collins, A. (July 1988). *The cognitive structure of emotions*. Cambridge University Press.
Pardos, Z. A., & Heffernan, N. T. (2010). Modeling individualization in a Bayesian networks implementation of knowledge tracing. In *Proceedings of User Modeling, Adaptation, and Personalization, (UMAP'2010): Vol. 6075 of lecture notes in computer science* (pp. 255–266). Springer.
Pearl, J. (1985). *Bayesian networks: A model of self-activated memory for evidential reasoning*. (UCLA Technical Report CSD-850017).
Pearl, J. (1986). Fusion, propagation, and structuring in belief networks. *Artificial Intelligence, 29*, 241–288.
Pearl, J. (1988). *Probabilistic reasoning in intelligent systems: Networks of plausible inference*. Morgan Kaufmann Publishers.
Pearl, J. (June 2000). *Causality: Models, reasoning and inference* (2nd ed.). Cambridge University Press.
Pourret, O., Naïm, P., & Marcot, B. (2008). *Bayesian networks: A practical guide to applications*. Wiley Interscience.
Reye, J. (1996). A belief net backbone for student modelling. In *Proceedings of International Conference on Intelligent Tutoring Systems (ITS'96). Vol. 1086 of lecture notes in computer science* (pp. 596–604). Springer.
Reye, J. (1998). Two-phase updating of student models based on dynamic belief networks. In *Proceedings of International Conference on Intelligent Tutoring Systems (ITS'98). Vol. 1452 of lecture notes in computer science* (pp. 274–283). Springer.
Reye, J. (2004). Student modeling based on belief networks. *International Journal of Artificial Intelligence in Education, 14*, 1–33.
Russell, S., & Norvig, P. (December 2002). *Artificial intelligence: A modern approach* (2nd ed.). Prentice Hall.
Self, J. (1988). Bypassing the intractable problem of student modelling. In: Proceedings of International Conference on Intelligent Tutoring Systems (ITS'98). pp. 107–123.
Self, J. (1994). Formal approaches to student modeling. In J. E. Greer, & G. I. McCalla (Eds.), *Student modelling: The key to individualized knowledge-based instruction. Vol. 125 of. Nato ASI Series F: Computer and systems sciences* (pp. 295–352). Springer Verlag.
Self, J. (1999). The defining characteristics of intelligent tutoring system research: ITSs care, precisely. *International Journal of Artificial Intelligence in Education, 10*, 350–364.
Stacey, K., Sonenberg, L., Nicholson, A., Boneh, T., & Steinle, V. (2003). A teaching model exploiting cognitive conflict driven by a bayesian network. In*Lecture notes in computer science, Vol. 2702* (pp. 145).
Suebnukarn, S., & Haddawy, P. (September 2006). Modeling individual and collaborative problem-solving in medical problem-based learning. *User Modeling and User-Adapted Interaction, 16*(3–4), 211–248.
Ting, C.-Y. T., & Phon-Amnuaisuk, S. (2009). Log data approach to acquisition of optimal bayesian learner model. *American Journal of Applied Sciences, 6*(5), 913–921.
VanLehn, K., Niu, Z., Siler, S., & Gertner, A. S. (1998). Student modeling from conventional test data: a Bayesian approach without priors. In*Lecture notes in computer science, Vol. 1452* (pp. 434–443). Berlin, Heidelberg: Springer Berlin Heidelberg.
Weisberg, M. (2007). Who is a modeler? *British Journal for Philosophy of Science, 58*, 207–233.
Wenger, E. (1987). *Artificial intelligence and tutoring systems: Computational approaches to the communication of knowledge*. Los Altos, CA: Morgan Kaufman.
Yudelson, M. V., Medvedeva, O. P., & Crowley, R. S. (2008). A multifactor approach to student model evaluation. *User Modeling and User-Adapted Interaction, 18*(4), 349–382.
Zapata-Rivera, J. D. (October 2004). Inspectable bayesian student modelling servers in multi-agent tutoring systems. *International Journal of Human-Computer Studies, 61*(4), 535–563.
Zukerman, I., & Albrecht, D. W. (March 2001). Predictive statistical models for user modeling. *User Modeling and User-Adapted Interaction, 11*(1), 5–18.