Chapter 4

Decision Analysis and Tutoring Strategy

The most prominent problem in the Intelligent Tutoring System is providing adaptive feedback to the students when the student needs feedback or help from the tutor. Feedback should be given in an adaptive manner. The issue of how much feedback to be provided to the student is also a complex one. Too little feedback will lead to frustrations and flounders (Anderson, 1993) while too much feedback may interfere in the learning process (Kashihiara et al., 1994). Once the amount of feedback to be given is finalized, choosing the content of feedback is a complex task.

Different teaching strategies used by teachers have been identified in the educational research (Kearsley, 2000). Most popular strategies are Spiral teaching (Bruner, 1992; Boehm et al., 1997) method and Socratic Method (Garkilov, 2000). In the curriculum, students tend to review the subjects they have already learned. Spiral teaching method takes this fact into consideration. Socratic Method is used to structure discourse between the tutoring system and the student. Some dialogue-based Intelligent Tutoring Systems are SCHOLAR designed by Carbonell (1970), WHY designed by Collins et al. (1996) and CIRCSIM-Tutor designed by Glass (2001).

Planning of strategy and its implementation have proven to be a difficult problem in Intelligent Tutoring System. Most of the Intelligent Tutoring Systems will not explicitly identify the strategies they are using for tutoring but implement some of the well known strategies (Major & Reichgelt, 1992). A better approach would be to specify the strategy
explicitly suitable for the domain, and to use the student model to select an appropriate action from the list of actions based on decision.

Introduction to decision theory and the construction of decision network from Bayesian Network are explained in detail in section 4.1. In section 4.2, the decision involved in BiTutor is discussed with multi-attribute utility function and the approach to augment the Bayesian Network to form decision network for action selection is explained. In section 4.3, the item selection is explained along with the methods by which items are attached to the Bayesian network. Finally, in section 4.4, the construction of Dynamic Decision Bayesian Network from Decision Network is explained and framing the strategy using this network is discussed.

4.1 Decision Theory
In this section, decision theory (Horvitz et al., 1988; Savage, 1954) is briefly introduced. While Bayesian networks are used to update beliefs from initial beliefs and observations, decision theory is a rational means of optimizing behavior by “fusing” uncertain beliefs with preferences. Suppose a rational agent is faced with the problem of selecting a single action from a set of possible actions \( D = \{d_1, d_2, \ldots, d_q\} \). (It is possible that one of the actions is the decision to perform no action.) If \( X = \{x_1, x_2, \ldots, x_n\} \) represents the possible outcomes of \( D \), then decision theory requires the agent to have a real-valued preference \( U(X, D) \) defined for each combination of values that \( X \) and \( D \) can take where \( P(X|D) \) is non-zero. \( U(X,D) \) is known as the utility function, and it is assumed that the agent’s preferences can in fact be translated into such a numeric form. Sometimes it is more convenient to encode the agent’s preferences along more than one dimension – this is permitted and is known as the multi-attribute utility (Mayo, 2001).
4.1.1 Utilities

Utility is giving preference between the different possible outcomes of the available actions. Utility theory provides a way to represent and reason with preferences. A utility function quantifies preferences, reflecting the "usefulness" of the outcomes, by mapping them to the real numbers. These mappings make the combination of utility theory with probability theory possible. This mapping can be used to find the action which is expected to deliver the most value (or utility) as its expected utility:

\[
EU(d_i) = \sum_{j=1}^{n} P(X_j \mid D_i) U(X_j, D_i)
\]  

The Principle of maximum expected utility states that the agent should select the action \( D = d_i \) that maximizes this expected utility. In generating utilities, the choice of unit is arbitrary, as the same decisions can result from the utility functions differing only in scale.

Two graphical methods are available in literature for representation of a decision problem namely, Decision Network and Decision Tree. In the following section, these two methods are briefly discussed.

4.1.2 Decision Network

A decision network is an extension of Bayesian Network using which the main considerations of decision making can be represented: the state of the world, the decisions or actions under consideration, the states that may result from an action and the utility of those resultant states. A decision network consists of three different types of nodes as shown in Figure 4.1.
Chance nodes: These are represented as oval shaped objects and are used to represent random variables exactly as in Bayesian Network. They can have decision nodes or other chance nodes as parents.

Decision nodes: These are represented as rectangular shaped objects and are used to represent the decision being made at a particular point of time. The values of a decision node are the actions that the decision maker must choose between. They can have chance nodes or other decision nodes as parents.

Utility nodes: These are represented as diamond shaped objects and are used to represent the agent’s utility function. Those variables describing the outcome state and directly affecting the utility are the parents of a utility node. Parents of a utility node may also include decision nodes. Each utility node has an associated utility table with one entry for each possible instantiation of its parents, perhaps including an action taken. When there are multiple utility nodes, the overall utility is the sum of the individual utilities.

4.1.3 Decision Tree

A decision problem can be represented by a decision tree. A decision tree consists of nodes connected by branches. There are three types of nodes in a decision tree: decision nodes, chance nodes, and utility nodes. The non-leaf nodes in a decision tree are either decision nodes or chance nodes and the leaf nodes are utility nodes. The nodes are represented using...
the same shapes as in decision network. Branches emanating from decision nodes represent action alternatives $D_i$. Branches emanating from chance nodes represent states $X_j$. Utilities $U(X_j, D_i)$ are placed at the ends of the appropriate branches. Probabilities $p_j$ are usually placed under the relevant chance branch.

Decision trees are drawn from left to right following the chronological sequence of the decision problem. Nodes need not necessarily alternate: any node may be followed by a similar or a different node; it all depends on the nature of the problem. Figure 4.2 shows the simplest case of a single decision $D$ followed by a single outcome $X$.

![Decision tree diagram](image)

**Fig 4.2: A decision tree for a single decision.**

**Example**

A decision network for a simple decision problem in tutoring system is shown in Figure 4.3 (a), let $C$ be the set of possible outcomes where their utility values are computed at the random variable $(KN)$, and the outcome of knowledge node $(KN)$ is dependent on the outcome of the student’s response $(R)$. The preference value $(U)$ of the tutoring action $(D)$ is
dependent on the variables \((KN)\) and \((R)\), but the decision is made without any information on the variable \((KN)\).

Figure 4.3 (b) shows the decision tree for the simple decision in tutoring system, the random variable \(KN\) is modeled with three discrete states, namely \(NM\), \(PM\), and \(M\), and decision \(D=\{d_1,d_2\}\) can take two different actions. Assuming, a student gives his response \(R\), the preference value for each action for the given response is also represented. In summary, the expected utility value at \(d_1\) is \(P(NM|R,d_1)U(c_1) + P(PM|R,d_1)U(c_2) + P(M|R,d_1)U(c_3)\), while the expected utility value at \(d_2\) is \(P(NM|R,d_2)U(c_4) + P(PM|R,d_2)U(c_5) + P(M|R,d_2)U(c_6)\), where \(c_i \in C, i=1,2,3,4,5,6\). The optimal action is generally chosen which aims to maximize the utility value at \(D\).

![Decision network and decision tree](image.png)

**Fig 4.3**: (a) Decision network for a simple decision problem.  
(b) Decision tree representation of the simple decision problem.
4.2 Decision Network for Bayesian Student Model

In BiTutor, Decision Network is an extension of the Bayesian Student Model with two types of nodes: utility nodes and decision nodes added. Each knowledge node in the Bayesian Student Model has an association with the utility variable (node). There should be arcs drawn from all the knowledge nodes to the utility node. Similarly, the decision node is also connected to the utility variable. There should be a single arc drawn from the decision node to the utility node. But, there will be no link between the knowledge nodes and the decision node as decision is made without any information from the knowledge node. That means the utility node is dependent on the knowledge nodes and the decision node. The Decision Network, thus formed, will look very complex with so many arcs to the utility nodes. In order not to encumber the network with too many arcs, the Bayesian Network is enclosed in a rectangle and only a single arc is drawn from the enclosed Bayesian Network to the utility variable as shown in Figure 4.4.

4.2.1 Utility Function in BiTutor

Utility is the main aspect in decision analysis (Howard & Matheson, 1983) to compare actions. It models decision makers’ preferences and values (Simon, 1960; Bernardo & Smith, 2000). In BiTutor, different possible actions would be selecting the best concept to teach; presenting a sub-topic to student; terminating the tutoring process if the student has achieved the required mastery of the topic; terminating the tutoring process if the student is not showing any progress in mastering the topic; terminating the tutoring process if there exists a pre-defined time-out or the student chooses to exit; terminating the tutoring process if the student is too dependent on the tutorials; selecting the best question used for testing the student (based on the mastery knowledge of the student); selecting the suitable feedback.
for the student if the student gives wrong response. In this research, the attributes associated with these actions are taken into considerations and are discussed in the following sections.

$$H(KN) = \sum_{i=1}^{k} P_i \times \log_2 P_i$$  \hspace{1cm} (4.2)$$

where $p$ is the proportion of $KN$ belonging to class $i$. Entropy can be viewed as a measure of the uniformness of a distribution and has a maximum value when $p_i = 1/K$ for all $i$. The goal is to have a peaked distribution of $P(KN)$ and to next select the best concept that has the smallest expected reduction in entropy, i.e.

Fig 4.4: Decision network for the topic on “Stack”.

Gain of Information

In BiTutor, the choice of the next best concept to be taught is made based on the Information Gain Theory. The commonly used measure of information from information theory (see Cover & Thomas, 1991), Shannon (1948) entropy, is applicable here:
$H(KN_0) - H(KN_j)_{i=1,2,...,n}$

where $H(KN_0)$ is the current entropy and $H(KN_j)$ is the expected entropy after administering knowledge node $i$, i.e. the sum of the weighted conditional entropies of the classification probabilities that correspond to the mastery and non-mastery of the knowledge node. The Gain of Information for tutoring the student on the knowledge node $KN^k$ except the topic node $KN'$ is defined as:

$$H(KN') = P(KN^k = NM)H(KN' | KN^k = NM) + P(KN^k = M)H(KN' | KN^k = M) \quad (4.3)$$

This can be computed using the following steps:

- Compute the normalized posterior classification probabilities.
- Compute the conditional entropies.
- Weigh the conditional entropies by their probabilities.

The utility value for the decision to tutor the knowledge node $KN^k$ is defined as:

$$u_i(c^k) = \begin{cases} 
   H(KN^k) & H(KN^k) \leq 1 \\
   H(KN^k) - \text{floor}(H(KN^k)) & H(KN^k) > 1 
\end{cases} \quad (4.4)$$

Gain of information for all nodes except the topic node in the Bayesian network is calculated, and the node with the lowest value indicates the best knowledge node, i.e. more information is gained by selecting the knowledge node with the lowest value. This knowledge node will be the best node for tutoring action.

A limitation with this method is that when the size of Bayesian network increases, the computation for $u_i(c^k)$ becomes inefficient. To solve this problem, two methods are available. Both these methods assume that the student will not forget a concept after mastering it.
**Method-I:** In Bayesian Student Model, as concept nodes are used as the smallest unit of knowledge, and this will be the node that contains the tutorial, then it is enough to calculate the gain of information (based on Equation (4.4)) for the concept nodes only. So, the concept node with the lowest value is selected for tutoring action. After the student has mastered the chosen concept, the gain of information of the remaining concept nodes has to be calculated. One drawback of this method is that the gain of information of all the concept nodes that are not mastered has to be calculated before each and every decision is made.

Table 4.1 shows the gain of information for the concept node “Push & Pop Operations” C4 on “Stack” topic. Using a similar procedure, the gain of information is calculated for all the concept nodes and the values are listed in Table 4.2. The C10 “Singly Linked List” concept has the lowest value of gain of information among all the concept nodes, and so C10 is the best concept to teach the student.

**Table 4.1:** Gain of information for the concept node “Push & Pop Operations” C4 on “Stack” topic.

| C4 | P(Nk|C4) | Conditional Entropy | P(C4) | H(KN) |
|----|--------|---------------------|-------|-------|
| NM | 0.529  | 1.462857            | 0.3   |       |
|    | 0.266  |                     |       |       |
|    | 0.205  |                     |       |       |
| PM | 0.179  | 1.343547            | 0.5   | 0.723638 |
|    | 0.616  |                     |       |       |
|    | 0.205  |                     |       |       |
| M  | 0.179  | 1.423905            | 0.2   |       |
|    | 0.266  |                     |       |       |
|    | 0.555  |                     |       |       |

**Table 4.2:** Gain of information for all concept nodes on “Stack” topic based on Method-I.

<table>
<thead>
<tr>
<th>KN</th>
<th>C4</th>
<th>C8</th>
<th>C9</th>
<th>C10</th>
<th>C11</th>
<th>C12</th>
<th>C14</th>
<th>C15</th>
<th>C16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gain</td>
<td>0.7236</td>
<td>0.6274</td>
<td>0.9214</td>
<td>0.4683</td>
<td>0.9181</td>
<td>0.9332</td>
<td>0.9319</td>
<td>0.9319</td>
<td>0.9632</td>
</tr>
</tbody>
</table>
Method-II: This method follows a greedy approach where the gain of information is calculated at each level and the best node for that level is chosen. This method calls for using a stack data structure and a queue data structure. The best node chosen at a level is pushed into the stack (using recursion). Then, the gain of information is calculated for all the children of the chosen node by traversing down the network using the chosen node and the best one again is pushed into the stack. The process is continued until a concept node is encountered. Whenever a concept node is encountered, it has to be inserted into the queue and should be popped from the stack and the other children of its parent should be considered. The process is continued until all the concept nodes are inserted into the queue. Figure 4.5 shows the algorithm for this method.

```
Gain (Parent, Child[]) {
    Calculate gain of information for all the children nodes that are not in queue based on the parent using Equation 4.4:
    NewParent = find the child with the smallest gain of information value;
    If (NewParent is a concept node) {
        Add NewParent to Queue;
        Gain(NewParent, Child[]);
    } Else {
        Ch [] = find all the children of NewParent;
        Gain(NewParent, Ch[]);
    }
}
```

Fig 4.5: Algorithm for finding the order of tutoring the concepts.

This method solves the problem associated with the Method-I. In this method, the gain of information is calculated only once for the entire network. The gain of information is calculated for all the concepts before making the first decision and the order of tutoring the concepts is maintained in a queue. The order of concepts in the queue gives the tutoring order of the concepts. After the student masters a concept, the next concept is taken from the queue. This method avoids the multiple calculations of gain of information. Table 4.3
shows the gain of information for the concept nodes of “Stack” topic computed using this method and also the order of tutoring the concepts is shown in that table.

Table 4.3: Gain of information on “Stack” topic based on Method-II using the algorithm (Figure 4.5).

<table>
<thead>
<tr>
<th>Parent</th>
<th>Children not in Queue</th>
<th>Gain Value</th>
<th>Selected Node</th>
<th>Stack</th>
<th>Queue</th>
</tr>
</thead>
<tbody>
<tr>
<td>C_1</td>
<td>C_2</td>
<td>0.520801</td>
<td>C_2</td>
<td>C_1</td>
<td>C_4</td>
</tr>
<tr>
<td></td>
<td>C_3</td>
<td>0.654164</td>
<td>C_4</td>
<td>C_1,C_2</td>
<td>C_4</td>
</tr>
<tr>
<td></td>
<td>C_5</td>
<td>0.875682</td>
<td>C_6</td>
<td>C_1,C_2,C_5,C_6</td>
<td>C_4</td>
</tr>
<tr>
<td>C_2</td>
<td>C_3</td>
<td>0.875682</td>
<td>C_3</td>
<td>C_1,C_2,C_3</td>
<td>C_4</td>
</tr>
<tr>
<td></td>
<td>C_8</td>
<td>0.771312</td>
<td>C_8</td>
<td>C_1,C_2,C_5,C_6</td>
<td>C_4</td>
</tr>
<tr>
<td>C_3</td>
<td>C_4</td>
<td>1.058785</td>
<td>C_9</td>
<td>C_1,C_2,C_3,C_5,C_6</td>
<td>C_4</td>
</tr>
<tr>
<td></td>
<td>C_9</td>
<td>0.613638</td>
<td>C_9</td>
<td>C_1,C_2,C_3,C_5,C_6</td>
<td>C_4</td>
</tr>
<tr>
<td>C_4</td>
<td>C_5</td>
<td>0.761065</td>
<td>C_5</td>
<td>C_1,C_2,C_3,C_5,C_6</td>
<td>C_4</td>
</tr>
<tr>
<td></td>
<td>C_6</td>
<td>0.761065</td>
<td>C_6</td>
<td>C_1,C_2,C_3,C_5,C_6</td>
<td>C_4</td>
</tr>
<tr>
<td>C_5</td>
<td>C_7</td>
<td>0.058785</td>
<td>C_7</td>
<td>C_1,C_2,C_3,C_5,C_6</td>
<td>C_4</td>
</tr>
<tr>
<td></td>
<td>C_10</td>
<td>0.941055</td>
<td>C_11</td>
<td>C_1,C_2,C_3,C_5,C_6</td>
<td>C_4</td>
</tr>
<tr>
<td></td>
<td>C_11</td>
<td>0.708144</td>
<td>C_11</td>
<td>C_1,C_2,C_3,C_5,C_6</td>
<td>C_4</td>
</tr>
<tr>
<td>C_6</td>
<td>C_10</td>
<td>0.941055</td>
<td>C_10</td>
<td>C_1,C_2,C_3,C_5,C_6</td>
<td>C_4</td>
</tr>
<tr>
<td></td>
<td>C_11</td>
<td>0.960783</td>
<td>C_11</td>
<td>C_1,C_2,C_3,C_5,C_6</td>
<td>C_4</td>
</tr>
<tr>
<td></td>
<td>C_12</td>
<td>0.711288</td>
<td>C_12</td>
<td>C_1,C_2,C_3,C_5,C_6</td>
<td>C_4</td>
</tr>
<tr>
<td></td>
<td>C_13</td>
<td>0.960783</td>
<td>C_13</td>
<td>C_1,C_2,C_3,C_5,C_6</td>
<td>C_4</td>
</tr>
<tr>
<td>C_7</td>
<td>C_14</td>
<td>0.908892</td>
<td>C_14</td>
<td>C_1,C_2,C_3,C_5,C_6</td>
<td>C_4</td>
</tr>
<tr>
<td></td>
<td>C_15</td>
<td>0.908892</td>
<td>C_15</td>
<td>C_1,C_2,C_3,C_5,C_6</td>
<td>C_4</td>
</tr>
<tr>
<td></td>
<td>C_16</td>
<td>0.940391</td>
<td>C_16</td>
<td>C_1,C_2,C_3,C_5,C_6</td>
<td>C_4</td>
</tr>
<tr>
<td>C_8</td>
<td>C_16</td>
<td>0.940391</td>
<td>C_16</td>
<td>C_1,C_2,C_3,C_5,C_6</td>
<td>C_4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Readiness to Terminate Tutoring Process

The utility function on readiness to terminate the tutoring process is defined as the maximum of the utility values stated in

$$u_j(R) = \max_{s=1,2,3,4}\{u_{2,j}(R)\} \quad (4.5)$$

The reasons for terminating the tutoring process could be that the student has achieved the required mastery of the topic or the student is not showing any progress in mastering the topic, a pre-defined time-out or the student chooses to exit, or the student is too dependent on the tutorials.
Mastery of a particular topic: Whenever a student answers a question, through evidence propagation, all the mastery states of the relevant knowledge nodes are updated. If the student has full mastery of the topic (a topic is said to be mastered if all the concepts in that topic are mastered), the utility value of the “readiness to terminate the tutoring process” attribute is defined as \( u_{2,1}(R) = 1 \). Otherwise, the measure of how close the student is to the desired mastery value state is defined in terms of the ratio of number of the concepts mastered to the total number of concepts in the topic: \( u_{2,1}(R) = \frac{n}{\zeta} \) where \( n \) is the number of concepts the student has mastered, and \( \zeta \) is a total number of concepts in the topic.

No Progress: Given the student is not progressing, the utility value of the “readiness to terminate the tutoring process” attribute is defined as: \( u_{2,2}(R) = \frac{n}{\mu} \) where \( n \) is the number of items the student has attempted that tests the concept, and \( \mu \) is a normalizing constant.

Time-out: Given the pre-defined number of actions is reached or the student chooses to exit the tutoring session, the utility value of the “readiness to terminate the tutoring process” attribute is defined as: \( u_{2,3}(R) = 1 \).

Too Dependent on Tutorials: Given the student is too dependent on tutorials, the utility value of the “readiness to terminate the tutoring process” attribute is defined as: \( u_{2,4}(R) = \frac{n}{\nu} \) where \( n \) is the number of times the student refers to the tutorial and \( \nu \) is a normalizing constant.
Multi-Attribute Utility Function

In BiTutor, two utility functions are defined. The first utility function is related to the gain of information, and the second utility function is related to the termination of tutoring. These two utility functions are mutually independent of each other. Keeney and Raiffa (1976) have demonstrated that, if the attributes are mutually independent, then the Multi-Attribute Utility Function (MAUF) can be written as follows:

\[ U(c_1, c_2) = k_1u_1(c_1) + k_2u_2(c_2) + k_1 k_2u_1(c_1)u_2(c_2) \]  

(4.6)

where \( U \) and \( u_1, u_2 \) are normalized to be bounded between zero (for the worst possible value) and one (for the best possible value) and \( k \) is a scaling constant that must satisfy the normalizing constraint \( 1 + k = \prod_{i=1}^{2} (1 + kk_i) \). If \( \sum_{i=1}^{2} k_i = 1 \) and, as a consequence, \( k \neq 0 \), the general utility function proposed by Keeney and Raiffa can be written in the following multiplicative form:

\[ kU(c_1, c_2) = \prod_{i=1}^{2} [kk_iu_i(c_i) + 1] \]  

(4.7)

On the other hand, if \( \sum_{i=1}^{2} k_i = 1 \), then \( k = 0 \), and the Keeney and Raiffa function converges to the following linear form:

\[ U(c_1, c_2) = k_1u_1(c_1) + k_2u_2(c_2) \]  

(4.8)

where \( 0 \leq k_i \leq 1, i = 1,2 \).

4.2.2 Action Selection

In decision-theoretic system, an action with the highest expected utility value is selected from the set of alternatives. Probability assignments of the random variables are separated from values computation and so the method is normative.
The expected utility for the tutoring action is defined as:

\[ U(a \mid KN') = \sum_{KN'} P(KN' \mid E, a) \times U(c_y) \]  

(4.9)

where:

- \( a \) denote the tutoring action that produces an outcome;
- \( KN' \) denote the \( y \)-state of the Bayesian network;
- \( U(c_y) \) denote the multi-attribute utility function defined in Equation (4.8); and
- \( P(KN' \mid E, a) \) denote the probability of attaining the \( y \)-state Bayesian network given the set of evidence \( (E) \) and the tutoring action \( (a) \).

### 4.3 Item Selection

In most of the Intelligent Tutoring System, evaluation of student's performance is done by matching his responses to the correct answers through items such as multiple-choice questions. Using multiple-choice questions, the question can be designed such that each item tests one concept. Selection of the best item is an important problem in Intelligent Tutoring System. For a student with low knowledge on the topic, items with high difficulty level should not be given. Similarly, for a student with high knowledge, items with low difficulty level should not be given. Item selection should be done based on the student knowledge. In this section, attachment of the item to the Bayesian network and selection of the best item using the Item Response Theory is explained in detail.

There are two ways where the Bayesian network can be used to select the suitable items. Millán et al. (2000) have adopted the first approach and this approach includes all the applicable items in a Bayesian network. This approach follows a static discipline where all the items are attached to the network well in advance. Figure 4.6 shows a decision network with four knowledge nodes (topic, subtopic node and two concepts nodes) and five items related to concepts nodes. This approach has a main drawback that if the number of items
increases, then the complexity of the decision network makes it very difficult to provide the conditional probability values and the utility values associated with the random variables.

Pek and Poh (2000) have adopted the second approach where the items that are used to collect evidence are not directly included in the Bayesian Network as shown in Figure 4.7a. If the decision is to assess the student on his mastery of a particular concept, then each item from the item bank is dynamically connected to the Bayesian Network (Figure 4.7b) to determine if it is suitable. Usually, an item is selected only when its difficulty value is marginally higher than the mastery state of the key concept attained by the student.
In BiTutor, the second approach is followed. Using the second approach, the construction of the network is much easier since not all the items need to be included in the computation of utility values. Figure 4.8 shows how the variable on student’s likely response is related to the topic of the Bayesian Network and the Mem parameter. The difficulty variable of the item is independent of the student’s mastery state of the topic. In this research, the probability of the student \((j)\) answering an item correctly \((x_{ji} = 1)\) \((i)\) is expressed using the concepts of Item Response Theory (IRT).

---

**Fig 4.7:** (a) Decision network for KN selection. (b) Selection of an item.
4.3.1 Item Response Theory

Item Response Theory (IRT) is a statistical framework in which students can be described by a set of one or more ability scores that are predictive, through mathematical models, linking actual performance on test items, item statistics, and student abilities. Refer Linden and Hambleton (1997), Hambleton et al. (1991) for fuller explanations on IRT. Under the two-parameters (2PL) IRT model, the probability of the student giving a correct response to the given item $i$ is a logistic function of the student's true ability and three item parameters (Figure 4.9):

$$P(\text{correct}) = \frac{1}{1 + e^{-(\theta - b_i - a_i c_i)}}$$

where $\theta$ is the student's true ability, $b_i$ is the difficulty parameter, and $a_i$ and $c_i$ are the discrimination and guessing parameters, respectively.
Fig 4.9: Three examples of the 2PL item response function with slope $a = 0.2, 1, 2$ and difficulties $b = -1, 0, 1$.

- $a$, is called the discrimination index, and defines the slope of the curve at its inflection point, therefore $a$, denotes how well the question is able to discriminate between students of slightly different abilities.

- $b$, is called the difficulty degree, and defines the location of the curve's inflection point. The higher the value of $b$, the more difficult the question will be.

- $\theta$ theta is called the student ability. In BiTutor, the expected mastery value of the student on the topic will be used as student ability.

An item for testing the concepts in a topic is said to be suitable if the probability of the student answering correctly is marginally less than $1/2$. When a suitable item is selected, the random variable on the item and students' response is decoupled from Bayesian Network. The student's response is then drawn out and used to instantiate the mastery state of the relevant concept (as discussed in Chapter 3). Through evidence propagation, all the mastery
states of the variables within the Bayesian network are updated. Thereafter, the decision and the utility nodes can be added to the Bayesian network. The parameters used in the 2-logistic function must be calibrated using the student’s responses. Item calibration is explained in detail in Chapter 5.

4.4 Tutoring Strategy

Before taking actions, the complete information on the variables in the student model is not available and so the decision-theoretic tutoring is a Partially Observable Markov Decision Problem (PODMP) (Lane, 1989; Cassandra et al., 1994). In PODMP, decisions are made by projecting forward a sequence of possible actions and choosing the best one. This sequence of actions to be used by the tutor is called Tutoring Strategy. These actions tell the tutor what to do for any state that the student might be in.

In section 4.4.1, Dynamic Bayesian Network is introduced and its construction is explained in detail. In the next section 4.4.2, the construction of Dynamic Decision Network from dynamic Bayesian Network is explained and the generation of tutoring strategy from the dynamic decision network is framed out.

4.4.1 Dynamic Bayesian Network (DBN)

Bayesian and decision networks model relationships between variables at a particular point in time or during a specific time interval. Although a causal relationship represented by an arc implies a temporal relationship, Bayesian networks do not explicitly model temporal relationships between variables. And the only way to model the relationship between the current value of variable, and its past or future value, is by adding another variable with a different name. In decision networks, there is an ad hoc modeling of time, through the use of information and precedence links. When making a sequence of decisions that will span a
period of time, it is also important to model how the world *changes* during that time. It is important to represent and reason about change over time explicitly when performing such tasks as diagnosis, monitoring, prediction and decision making.

To determine the tutoring strategy, Dynamic Bayesian network must be constructed (Zweig and Russell, 1997). The simplest dynamic Bayesian network is formed when two Bayesian networks in consecutive time steps are connected together.

In BiTutor, the domain knowledge consists of a set of $n$ knowledge nodes $KN = \{KN_1, KN_2, \ldots, KN_n\}$, each of which is represented by a node in a Bayesian network as discussed in Chapter 3. When constructing a DBN for modeling changes over time, one node for each $KN_i$ is included for each time step. If the current time step is represented by $(t)$ and the next time step by $(t+1)$, then the corresponding DBN nodes will be

- Current: \(\{KN_1^{(t)}, KN_2^{(t)}, \ldots, KN_n^{(t)}\}\)
- Next: \(\{KN_1^{(t+1)}, KN_2^{(t+1)}, \ldots, KN_n^{(t+1)}\}\)

Each time step is called a *time-slice*. The relationship between knowledge nodes in a time-slice is represented by *intra-slice* arcs $KN_i^{(t)} \rightarrow KN_j^{(t)}$. Although it is not a requirement, the structure of the time-slice does not usually change over time. The relationships between the knowledge nodes $\{KN_1^{(t)}, KN_2^{(t)}, \ldots, KN_n^{(t)}\}$ are the same, regardless of the particular $T$. The relationships between knowledge nodes at successive time steps are represented by *inter-slice* arcs, and also called *temporal arcs*, including relationships between (i) the same knowledge node over time, $KN_i^{(t)} \rightarrow KN_j^{(t+1)}$, and (ii) different knowledge nodes over time, $KN_i^{(t)} \rightarrow KN_j^{(t+1)}$ (for more details see Neapolitan,

---

Decision Analysis and Tutoring Strategy 95
In BiTutor, only the relationship between the same knowledge nodes over time is considered.

Figure 4.10 shows the general structure of a dynamic Bayesian network, with a sequence of the same static Bayesian networks with inter-slice arcs shown as solid arcs. Note that there are no arcs that span more than a single time step. From Markov assumption, the state of the world at a particular time depends only on the previous state and any action taken in it.

The relationships between knowledge nodes, both intra-slice and inter-slice, are quantified by the conditional probability distribution associated with each node. For knowledge node $KN_i^{(T)}$ with intra-slice parents $KN_1^{(T)}, KN_2^{(T)}, \ldots, KN_n^{(T)}$ and inter-slice parents $KN_i^{(T-1)}$, the CPT is $P(KN_i^{(T)} | KN_1^{(T-1)}, KN_2^{(T-1)}, \ldots, KN_n^{(T-1)})$.

![General structure of a Dynamic Bayesian Network](image)

In dynamic Bayesian network, the number of arcs will increase, if the Bayesian network becomes complex, making the dynamic Bayesian network look clumsy. To avoid this problem, in BiTutor, each time slice is enclosed inside a rectangle and a single arc is
drawn from time slice $t$ to the next time slice $t+1$ representing the relationship between $KN_{i}^{(t)} \rightarrow KN_{j}^{(t+1)}$.

In BiTutor, the states of the knowledge nodes may change from one time-slice to another. This calls for the use of the formula stated in 3.5 (updating Bayesian network, Chapter 3) to determine the new mastery state after an action, as in Figure 4.11. The conditional probabilities $P(KN|pa(KN))$ and $P(E|KN)$ do not change with time.

![Dynamic Bayesian Network](image)

**Fig 4.11:** Dynamic Bayesian Network maintained as a sliding “window” of time-slice. (Shading indicates evidence node.)

Figure 4.12 shows a two-time slice based DBN for the “Stack” topic where the dashed arcs model the conditional probability distributions, $P(KN_{i}^{(t)}|KN_{i}^{(t-1)}, a^{(t-1)})$, which describes how the next states of the variables ($KN_{i}^{(t)}$) depend on the student’s response ($E_{i}^{(t-1)}$) to the action ($a^{(t-1)}$) of the agent. In BiTutor, $P(KN_{i}^{(t)}|KN_{i}^{(t-1)})$ is set to $(1 - \Theta \times k)$ where $\Theta$ is the rate of learning update and $k$ is an empirical constant. This probability is not set to 1 as to consider uncertainty such as carelessness, lucky guesses, and forgetting.
Fig 4.12: A two-slice of a Dynamic Bayesian Network for monitoring of student's mastery state on "Stack" topic where dashed arc represents $KN_i^{(T)} \rightarrow KN_j^{(T+1)}$. 
The shaded node in the DBN represents the variable of the present Bayesian network where evidence \((E^{t-1})\) of the student’s mastery has been simulated as a result of the tutoring action \((a^{t-1})\). Belief updating for each time slice, from the first time slice up to and including the current time slice \(t\), of DBN can be done using the standard Bayesian network inference algorithms. New posterior distributions can be obtained for all the non-evidence nodes, including nodes in the \(t+1\) and later time slices. This updating into the future is called **Probabilistic projection**.

**Probabilistic projection**

- Let \(KN_{t} \subseteq KN, E_{t} \subseteq E\) and \(a_{t} \in D\) at time \(t\).
- Assume at \(t = 0\) (beginning of tutoring), the action \(a^{(0)}\) is a pre-defined decision, such as presenting the key points on the chosen topic.
- The probability distributions of \(KN_{t}\) for different instantiations of evidence and actions up to time \(t\) are updated (Tatman & Shachter, 1990) using the prediction and estimation of new belief based on projected evidence (Russell and Norvig, 2003).

\[
\text{For } t = 1, 2, \ldots, n \text{ and } n < \infty,
\]

\[
P(KN_{t} | E^{(1)}, E^{(2)}, \ldots, E^{(t)}, a^{(0)}, a^{(1)}, \ldots, a^{(t-1)} )
= a \times P(E^{(1)} | KN^{(0)}) \\
\times P(KN^{(t)} | KN^{(t-1)}, a^{(t-1)})
\times P(KN^{(t-1)} | E^{(1)}, E^{(2)}, \ldots, E^{(t-1)}, a^{(0)}, a^{(1)}, \ldots, a^{(t-2)})
\]

where \(a\) is a normalizing constant

**Prediction**

**Estimation**

**Conditional probability distribution for other time steps**

4.4.2 Dynamic Decision Network (DDN)

Just as Bayesian network can be extended with a temporal dimension to give dynamic Bayesian network, so can decision network be extended to give dynamic decision network. Decision and utility nodes are included in dynamic Bayesian network to form the decision
dynamic network (Castillo et al., 1997). Not only do they represent explicitly how the world changes over time, but they also model general sequential decision making.

Once the evidence of student's mastery for the knowledge node (Concept) is given, the dynamic decision network is used to determine the tutoring strategy for POMDP. In the optimized tutoring strategy, the selected actions tend to maximize the student's likelihood of mastering the topic. An optimal tutoring strategy \( A^* \) is one which satisfies the maximum expected utility of all the actions in that strategy.

**Tutoring Strategy**

Let \( c^{(t)} \) be the outcome of the attributes due to the action taken at time \( t \). The maximum expected utility at the \( t^{th} \) decision is evaluated by a summation over all the random variables \( (KN^{(t)}) \) in the Bayesian network and a maximization over the alternative actions \( (D^{(t)}) \). At any time slice, \( t \) (where \( 1 \leq t \leq n \)),

\[
d^t = \max_{a^{(t)}} \left[ \sum_{KN^{(t)}} P(KN^{(t)} | E^{(t)}, ..., E^{(t)}, a^{(t)}, ..., a^{(t-1)}) \times U(c^{(t)}) \right] \forall a \in D
\]

The optimal tutoring strategy is the set of tutoring actions over \( n \) time slices:

\[ A^* = \{a^{(t)}, a^{(2)}, ..., a^{(n)}\} \]

Figure 4.13 is the graphical representation of the dynamic decision network that has been projected three steps into the future; the current and future decisions and the future observations are all unknown. \( U^{(t+3)} \) represents the utility values for the \( (t+3) \) time slice and all subsequent utility values. The objective is to maximize the sum of all future utility values.
Figure 4.14 shows part of the search tree corresponding to the three-step look-ahead DDN shown in Figure 4.13. Each of the triangular nodes is a belief state in which the tutor makes a decision \( A^{i,j} \) for \( i = 0,1,2, \ldots \). The round nodes correspond to what evidence \( E^{i,j} \) occurs. There are no chance nodes corresponding to the action outcomes; this is because the belief state update for an action is deterministic regardless of the actual outcome. For decision \( A^{i,j} \), the tutor will have available evidence \( E^{i-1,j}, E^{i+1,j}, \ldots, E^{i,j} \) even though at time \( t \) it does not know what those evidences will be. The tutor takes into account the values of information and will execute information-gathering actions where appropriate. A decision can be extracted from the search tree by backing up the utility values from the leaves, taking an average at the chance nodes and taking the maximum at the decision nodes (Russell and Norvig, 2003). Finding the optimal tutoring strategy will be very complex, if there is no approximation to \( f^{i,j} \).
This research applies the property of separability to the utility functions in order to find the approximate solution to the POMDP. A utility function is separable if and only if a function \( f \) can be found such that:

\[
U(c_0), c(1), \ldots, c(n) = f(c_0, U(c_1), c(2), \ldots, U(c(n)))
\]

where \( U(c_0) \) is the utility function for the outcome at time slice \( t \).

The simplest form of separable utility function is additive:

\[
U(c_0, c_1, \ldots, c(n)) = U(c_0) + U(c_1) + \ldots + U(c(n))
\]

The tutoring strategy can be written using a sequence of IF-THEN-ELSE statements. Conditions given in the IF statements are matched to the student's responses during the

---

Decision Analysis and Tutoring Strategy 102
tutoring and the appropriate actions are triggered. Then, the Bayesian network is updated using the student's responses. When all his responses have been gathered, the updated Bayesian network can be used to determine the next tutoring strategy. This process is continued until the termination decision is made. The duration of the tutoring session may vary depending on the student's responses.

While providing the tutoring strategy, the tutor should be able to foresee the required number of actions or to see the actions until the termination of tutoring criteria is met. Based on the initial EMV of the selected topic, the items are taken from the item bank to test the student using the concepts of Item Response Theory (IRT). This means choosing the items based on the mastery level of student on that concept. This makes the tutoring system adaptive. Based on the student's response to the selected item, Expected Mastery Value (EMV) is updated using the Equation (3.5) in Chapter 3. At this point, future actions are broken into two branches: sequence of actions if the student's response is correct, sequence of actions if the student's response is wrong. Based on the new state of the concept, the best item to test the concept is chosen and the process continues. If the student masters the concept, next best concept to teach is chosen and the student is tested with the items from that concept (with the appropriate level of difficulty).

Example

Figure 4.15 illustrates the tutoring strategy generated for a student who has been tutored on the “Stack” topic by BiTutor. The student is currently learning the “Push and Pop operations” $C_4$ concept of “Stack” $C_1$ topic. Initial EMV is calculated for $C_4$ $EMV(C_4) = 0.5175$. It is found that the student is in partial mastery state for that concept. As per IRT, an item to test $C_4$ is chosen from the level 3. $C_4$ moves to a new state based on the student's response. If the student's response is correct, $EMV(C_4) = 0.6675$, an item with difficulty
level 4 is chosen. If the student's response is wrong, $EMV(C_4) = 0.3675$, an item with
difficulty level 2 is chosen. The optimal actions are chosen for each subsequent states of $C_4$.

These sets of optimal actions form the tutoring strategy.

---

Initial $EMV(C_4) = 0.5175$ - Partial Mastery
Test $C_4$ with item from Level 3.
IF (Response is Correct)

1. Update $EMV(C_4) = 0.6675$ - Partial Mastery
Test $C_4$ with item from Level 4.
IF (Response is Correct)

2. Update $EMV(C_4) = 0.8675$ - Mastery
Choose $C_1$ with $EMV(C_1) = 0.5175$ - Partial Mastery
Test $C_1$ with item from Level 3.
IF (Response is Correct)

3. Update $EMV(C_1) = 0.6675$ - Partial Mastery
Continue until the termination of tutoring criteria is met.
ELSE-IF (Response is Wrong)

4. Update $EMV(C_1) = 0.3675$ Non Mastery
Continue until the termination of tutoring criteria is met.
ELSE-IF (Response is Wrong)

5. Update $EMV(C_4) = 0.5675$ - Partial Mastery
Test $C_4$ with item from Level 3.
IF (Response is Correct)

6. Update $EMV(C_4) = 0.7175$ - Partial Mastery
Continue until the termination of tutoring criteria is met.
ELSE-IF (Response is Wrong)

7. Update $EMV(C_4) = 0.4175$ - Partial Mastery
Continue until the termination of tutoring criteria is met.
ELSE-IF (Response is Wrong)

8. Update $EMV(C_4) = 0.3675$ Non Mastery
Test $C_4$ with item from Level 2.
IF (Response is Correct)

9. Update $EMV(C_4) = 0.4675$ - Partial Mastery
Test $C_4$ with item from Level 3.
IF (Response is Correct)

10. Update $EMV(C_4) = 0.6175$ - Partial Mastery
Continue until the termination of tutoring criteria is met.
ELSE-IF(Response is Wrong)
{
Update EMV(C4); EMV(C4) = 0.3175 - Non Mastery
Continue until the termination of tutoring criteria is met.
}
ELSE-IF(Response is Wrong)
{
Update EMV(C4); EMV(C4) = 0.1675 - Non Mastery
Test C4 with item from level 1.
IF (Response is Correct)
{
Update EMV(C4); EMV(C4) = 0.2175 - Non Mastery
Continue until the termination of tutoring criteria is met.
}
ELSE-IF(Response is Wrong)
{
Update EMV(C4); EMV(C4) = 0 - Non Mastery
Termination of tutoring criteria is met. Student has to leave the tutoring session.
}
}

Fig 4.15: Tutoring strategy with 3 actions.

Figure 4.16 shows a simpler version of tutoring strategy shown in Figure 4.15. In BiTutor, tutoring strategy is provided as this simpler version to staff (teachers).

Test C4 with item from level 3
If response is correct then
Test C4 with item from level 4
If response is correct then
Test C4 with item from level 3
Else
Test C4 with item from level 3
Else
Test C4 with item from level 2
If response is correct then
Test C4 with item from level 3
Else
Test C4 with item from level 3
Else
Test C4 with item from level 1

Fig 4.16: Simpler version of tutoring strategy with 3 actions.