Tutorial 1. Building a Bayesian Network

This tutorial shows you how to implement a small Bayesian network (BN) in the Hugin GUI. The BN you are about to implement is the one modelled in the apple tree example in the basic concepts section. In the next tutorial you will extend this BN to an influence diagram.

The qualitative representation of our BN is shown in figure 1.

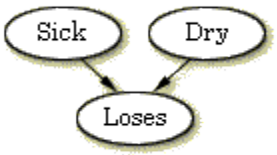

Figure 1: BN representing the domain of the apple tree problem.

If you want to understand the design of this BN you should read about it in the basic concepts section.

Constructing a New BN

When you start up the Hugin GUI, the edit window opens. This window contains a menu bar, a tool bar and a document pane. In the document pane, a new empty network called "unnamed1" is automatically opened in a network window (see figure 2). It starts up in "edit" mode, which allows you to start constructing the BN immediately (the other main mode is "run" mode which allows you to use the BN).

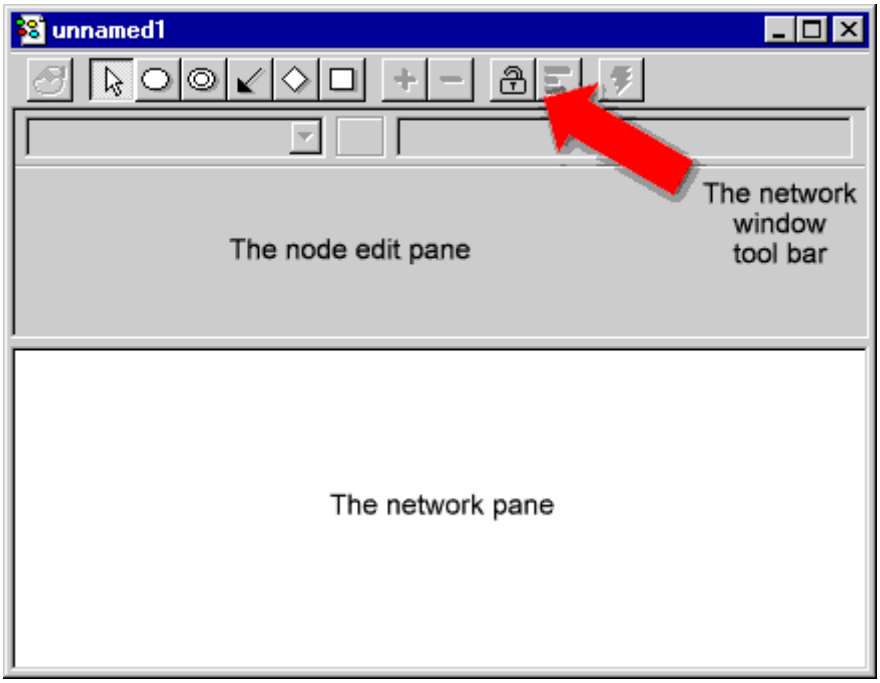

Figure 2: The network window containing a tool bar, a node edit pane, and a network pane.

Adding Nodes

The first thing we will do is add the Sick node. This can be done as follows:

- Select the discrete chance tool in the tool bar of the "unnamed1" network window (see figure 3)
- Click somewhere in the network pane (see figure 2)
-

When you have clicked in the network pane, the node "C1" appears. We want to change this label to "Sick":

- Select the node using the mouse
- Enter "Node Properties" by pressing the node properties tool (see figure 3)
- Change both "Name" and "Label" fields to "Sick"
- Press "OK"

The "Name" is the internal name of the node, while "Label" is the label of the node. If no label is specified (as was the case before you changed the label) the label used is the internal name. The internal name can consist of only the letters a-z and A-Z, the digits 0-9, and the underscore character _ whereas the label can be almost anything.

Figure 3: From left: The discrete chance tool, the node properties tool, and the causal arrow tool.

The Dry and Loses nodes are added the same way. You can add more nodes without having to press the discrete chance tool all the time by holding down the SHIFT key while clicking in the network pane. When you have chosen a node in the network pane you can access the node properties tool by holding down the right mouse button.

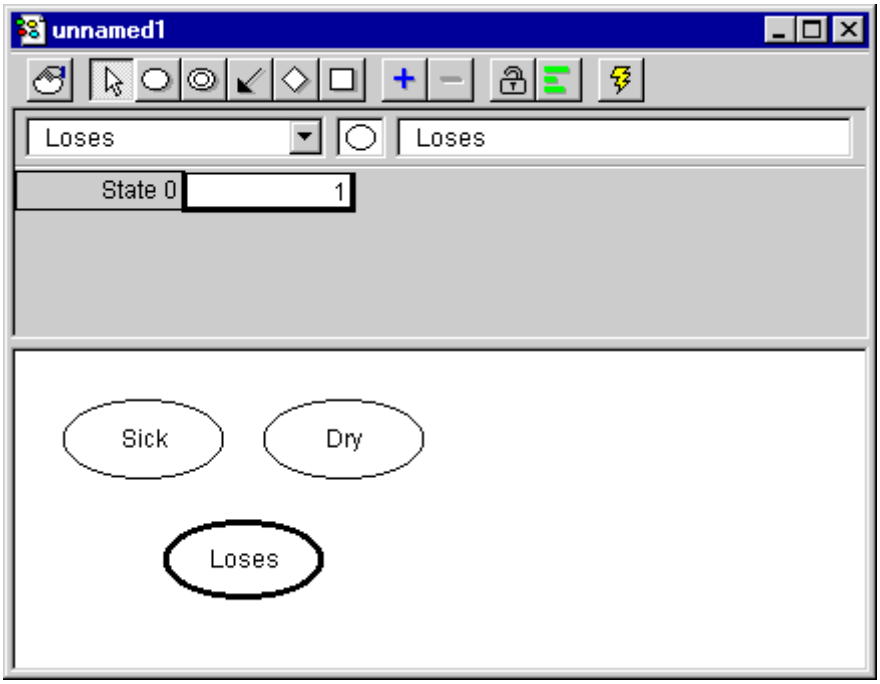

Figure 4: The network pane contains the three nodes Sick, Dry, and Loses that have been added to the BN. The node edit pane contains the CPT of the currently active node.

Adding Causal Arrows

Now, you should have a BN similar to the one shown in the network pane in figure 4. To add the causal arrows from Sick to Loses and from Dry to Loses, do as follows:

- Press the causal arrow tool (see figure 3)
- Drag an arrow from Sick to Loses with the left mouse button while holding down the SHIFT key. The SHIFT key ensures that you can add more arrows without having to press the causal arrow tool again
- Drag an arrow from Dry to Loses with the left mouse button

What you have by now should be the complete qualitative representation which is similar to the one in figure 1. The next step will be to specify the states and the conditional probability table (CPT) of each node.

The States

In the introduction to BNs the states of the nodes were specified as follows: Sick has two states: "sick" and "not", Dry has two states: "dry" and "not", and Loses has two states "yes" and "no".

First, we shall tell you how to specify the states of Sick:

- Choose the Sick node as the currently active node by selecting it from the drop down list below the tool bar or simply by double clicking it
- Press the add state tool in the tool bar (see figure 5)
- Click the field containing the text "State 0" in the cpt in the node edit pane
- Type the text "sick" in the field to give the state this name
- Click the field containing the text "State 1" in the CPT
- Type the text "not" in the field

Now, do the same with Dry.

Figure 5: The add/delete state tools.

You can do exactly the same with Loses, but you might be a little surprised when selecting Loses as the active node because the CPT of Loses is a little bigger than those of Sick and Dry. This is just because Loses has parent nodes (which Sick and Dry have not).

- Add a state to Loses and name the two states "yes" and "no".

Entering CPT Values

The next step is to enter the CPT values correctly (as default the Hugin GUI has given all nodes a uniform distribution). The values were specified in the introduction to BNs and they are shown in table 1, 2, and 3.

Sick="sick"	**Sick**="not"
0.1	0.9

Table 1: P(Sick).

Dry="dry"	**Dry**="not"
0.1	0.9

Table 2: P(Dry).

	Dry="dry"		**Dry**="not"	
	Sick="sick"	**Sick**="not"	**Sick**="sick"	**Sick**="not"
Loses="yes"	0.95	0.85	0.90	0.02
Loses="no"	0.05	0.15	0.10	0.98

Table 3: P(Loses | Sick, Dry).

First, enter the values into the Sick node:

- Choose the Sick node as the currently active node
- Click the field representing Sick="sick"
- Enter the value 0.1 (from table 1)
- Click the field representing Sick="not"
- Enter the value 0.9 (from table 1)

Enter the values of Dry and Loses the same way. When you enter the values into the CPT of the Loses node, be careful to get it done right. When you have entered all CPTs, the network window should look like figure 6.

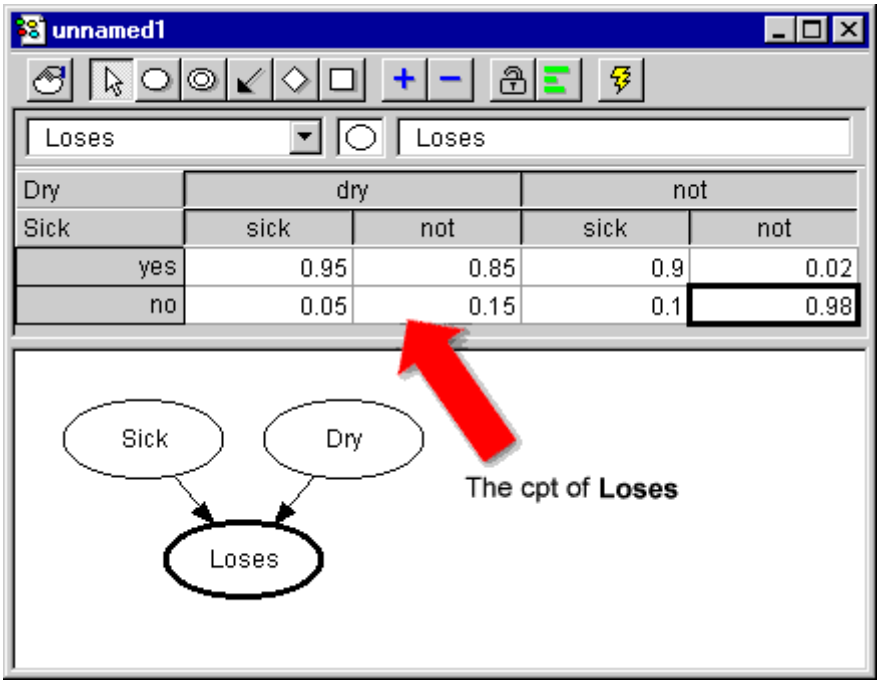

Figure 6: The document window with Loses chosen as currently active node. The CPT of Loses is seen in the node edit pane.

This finishes the construction of the BN. At this point it would be a good idea to save the BN. Here is how to do it:

- Select "Save As" from the "File" menu
- Enter a name (e.g. "apple.hkb")
- Press "Save"

Now we have finished constructing the Hugin knowledge base using Bayesian network technology. Now we want to compile and run the Hugin knowledge base and see if it is behaving correctly.

Building a Bayesian Network (Continued)

Compiling the BN

Now it is time to compile the network and see how it works:

Figure 7: The run mode tool button

The compiler checks for the following errors:

- Cycles. There must be no cycles in a network (whether or not there is a cycle does not depend on the directions of the arrows)
- For each parent configuration of a node the probabilities of the different states must have the sum of 1. In other words, each column of the table must sum to 1. If there is a column that does not sum to 1, the compiler will normalize the values. This fact can be utilised when filling in the node properties. Say, for example, that the probability of a tree being sick is based on the observation of 13527 trees over one season, where 1678 got sick and the rest didn't. Instead of first calculating the fractions, you just put 1678 in the sick state of the Sick node, and 11849 in the no state. The compiler will then calculate the proper values of probability

If you have followed the guidance of this tutorial, there should not be any errors in the compilation process. The compilation should be finished very fast with a small BN like ours. After the compilation, the "run" mode is entered (so far you have only been working in "edit" mode).

Running the BN

Running in "run" mode, the network window is split into two by a vertical separation (see figure 8). To the left is the node list pane and to the right is the network pane.

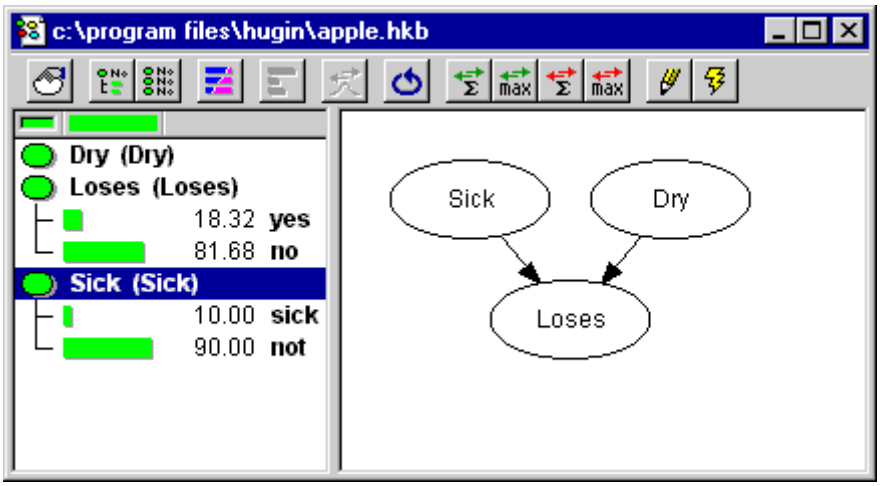

Figure 8: The network window in "run" mode. To the left is the node list pane (having Loses and Sick expanded) and to the right is the network pane.

You can view the probabilities of a node being in a certain state by expanding the node in the node list pane. You expand a node by double clicking it in the node list pane (if you click on the small node icon in the node list pane, you only need to click it once). Now, expand Loses and Sick:

- Double click Loses in the node list pane
- Double click Sick in the node list pane

You can also expand all nodes at once by pressing the expand node list tool in the tool bar just to the right of the node properties tool.

Is the tree sick?

Now imagine that you want to use your BN to find the probability of an apple tree being sick given the information that the tree is losing its leaves. This is done as follows:

- Expand all nodes (by pressing the expand node list tool)
- Enter the fact that the tree is losing its leaves by double clicking the state "yes" of the Loses node
- Propagate the BN by pressing the sum propagation tool in the tool bar (see figure 9)
- Read the probability of Sick being in state "sick"

Figure 9: The sum propagation tool

This should give the output shown in figure 10.

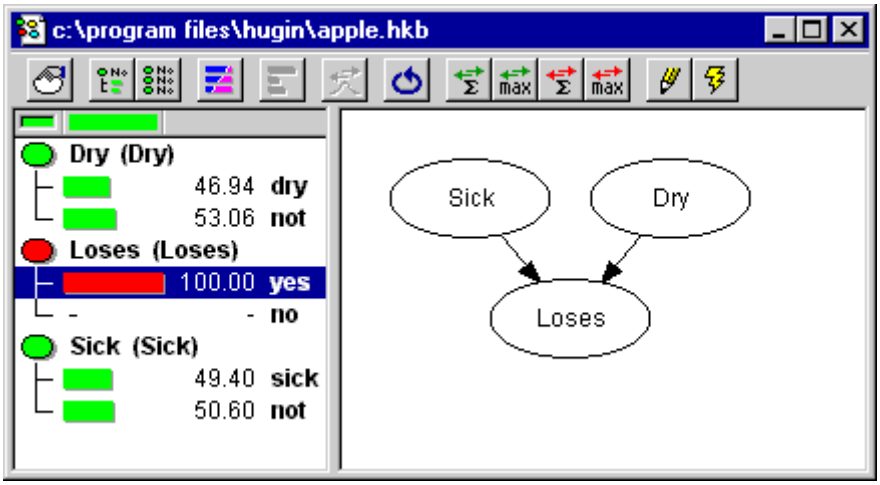

Figure 10: Our BN after entry of evidence that the tree is losing its leaves and sum propagation

The probability of the tree being sick is now 0.49.

If you do not read the value specified above, you have probably mistyped something when filling in the CPTs. Check the CPTs of all the nodes.

The Monitor Windows

In the last section, you used the node list pane to enter evidence and retrieve beliefs. You can also do this by using the monitor windows. The monitor windows show the same information as the node list pane, but you have the opportunity to place the monitor windows near the corresponding nodes of the BN in the network pane. You can open a monitor window for each node in the network pane, but the best way to use them is probably only to open a monitor window for the nodes in the BN which has special interest, otherwise they might take up too much space.

Now we shall open monitor windows for Sick and Loses and repeat the calculations from before. First, initialize the BN:

- Press the initialize tool button (to the left of the sum propagation tool)

Then we are ready to open the monitor windows of Sick and Loses.

- Select Sick and Loses (hold down the SHIFT key to select more nodes at the same time)
- Choose "Show Monitor Windows" from the "View" menu

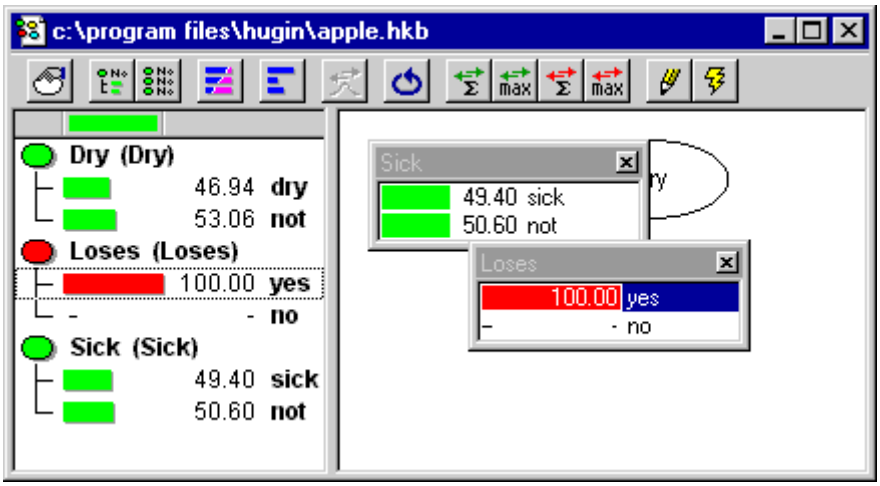

Figure 11: Monitor windows of Sick and loses are now shown in the network pane

Now you know all you need to know to continue to the second tutorial showing you how to construct a Hugin knowledge base using influence diagrams. The rest of this first tutorial introduces some useful aspects of the Hugin GUI, but can be skipped.

The most Likely Combination

From the propagation in the previous section we could see that the probability of the tree suffering from drought is 0.47. In both the case of Sick and Dry it is more likely that the state is "not". This could make one believe that the most likely combination of states is when both Sick and Dry are in state "not". However, this is a wrong conclusion. If you want to find the most likely combination of states in all nodes, you should use max propagation (instead of sum propagation). The max propagation tool is found on the tool bar just to the right of the sum propagation tool.

Now try to press the max propagation tool. In each node, a state having the value 100.00 belongs to a most likely combination of states. In this case, this gives one unique combination being the most likely: Sick is "sick" and Dry is "not"

We see that even if Sick="sick" is less likely than Sick="not", Sick="sick" is contained in the most likely combination of the states of the nodes while Sick="not" is not. This illustrates that you need to be careful about the conclusions you make from the result of a propagation.

Now one might want to know the probability of this most likely combination of states (or of any other combination of states) under the assumption that the entered evidence holds.

Calculating the Probability of a Combination of States

Here, we shall describe a technique to calculate the probability of the most likely combination of states given the evidence that the apple tree is losing its leaves. This probability is written:

$$P(\text{Sick="yes", Dry="not"} \mid \text{Loses="yes"})$$

Any time you perform sum propagation in your BN, the probability of the entered evidence is shown in the lower left corner of the HUGIN Runtime window (the P(All) value). If you have chosen the "yes" state of the Loses node and performed sum propagation, you can read the probability of Loses="yes" (written P(Loses="yes")). This value should be 0.1832.

The technique uses the following rule from probability theory (known as the fundamental rule):

$$P(A, B) = P(A \mid B)\, P(B)$$

The only kind of probability we can get from HUGIN is the probability of a series of bits of evidence which can be written in the form:

$$P(A1, A2, \ldots, An)$$

We use the fundamental rule to rewrite our requested probability to some expression composed by such components:

$$P(\text{Sick="yes", Dry="not"} \mid \text{Loses="yes"}) = P(\text{Sick="sick", Dry="not", Loses="yes"}) / P(\text{Loses="yes"})$$

In the fundamental rule, we have divided both sides with P(B). Then we have substituted A with Sick="yes", Dry="not" and B with Loses="yes".

We already know P(Loses="yes") so we only need to calculate P(Sick="sick", Dry="not", Loses="yes"). This is done as follows:

- Enter Sick="sick", Dry="not", and Loses="yes" in the BN
- Press the sum propagation tool
- Read P(Sick="sick", Dry="not", Loses="yes") as the P(All) value in the lower left corner

This value should be 0.081. Now, we are ready to calculate the requested probability:

P(Sick="yes", Dry="not" | Loses="yes") = 0.081 / 0.1832 = 0.442

Therefore the probability of the most likely combination of states of Sick and Dry, given that Loses="yes", is 0.442.

This finishes the first tutorial. If you want to learn how to construct a small influence diagram, go on to tutorial 2, Building a (Limited Memory) Influence Diagram.

Tutorial 2. Building a (Limited Memory) Influence Diagram

This tutorial shows you how to implement a small influence diagram in the Hugin GUI. It requires that you have already constructed the Bayesian network from the first tutorial. The influence diagram you are about to implement is the one modelled in the introduction to influence diagrams in the getting started section. It helps plantation owner Apple Jack to decide whether or not to give his apple tree, which is losing its leaves, some treatment. The qualitative representation of the influence diagram is shown in figure 1.

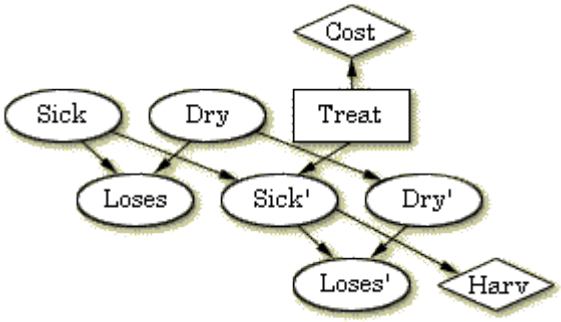

Figure 1: The qualitative representation of the influence diagram used for decision making in Apple Jacks plantation

Open the BN for Editing

First, you must open the BN constructed in the first tutorial if it is not already open. Here is how to do it:

- Select "Open" from the "File" menu
- Enter the name of the BN file ("apple.hkb"). You can do this by selecting it from the list of BN files (which have the "hkb" extension).

In figure 2, the BN has been opened and the Hugin GUI is currently working in "edit" mode. We need to be in "edit" mode to edit the BN, so if your network window shows the BN in "run" mode, press the "edit" mode tool button. If you opened it in "edit" mode, you do not need to do anything.

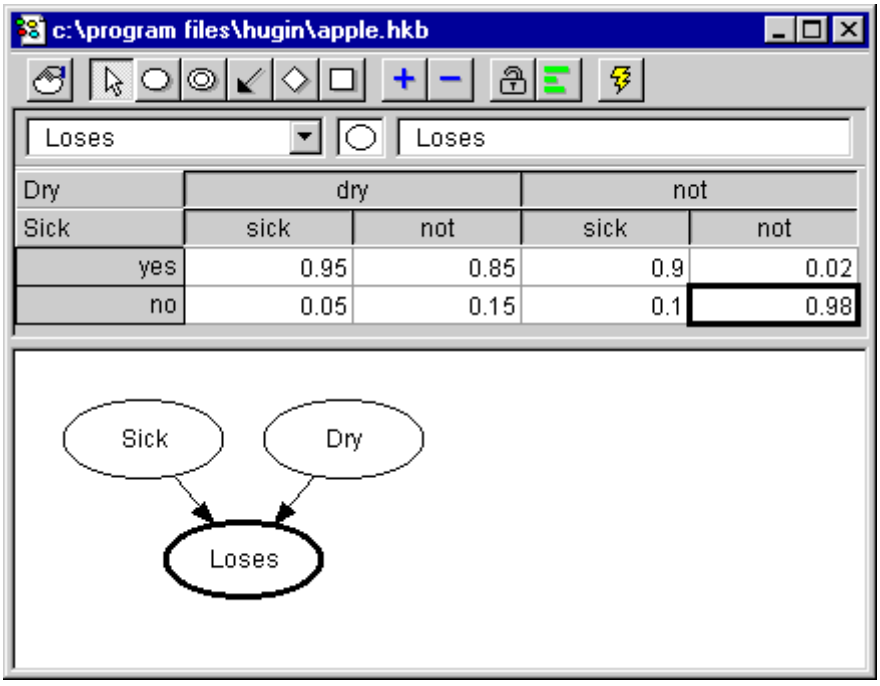

Figure 2: The open BN from the first tutorial in "edit" mode.

Copying Nodes

In the influence diagram in figure 1, there are three nodes very similar to those that we already have. In this case, the Hugin GUI allows you to copy a group of nodes and paste them in another area of the network pane. Here is how to do it:

- Create a rectangle selection with the mouse cursor around all three nodes (drag a rectangle by holding down the left mouse button)
- Select "Copy Nodes" from the "Edit" menu in the Hugin GUI window tool bar
- Select "Paste Nodes" from the "Edit" menu in the Hugin GUI window tool bar
- Move the new group of nodes to a spot where there is room for them

The Hugin GUI generates new names and labels for the new nodes. You can keep the names and change the labels to Sick', Dry', and Loses' (you cannot use "Sick'" as the name because it contains the prime character which is illegal in names). Do you remember how to change the labels?:

- Select the node with the mouse cursor
- Enter "Node Properties" by pressing the node properties tool (the left most tool button in the tool bar of the network window)
- Change the "Label" field
- Press the "OK" button

You perform the steps above for all three new nodes. Your network should then look as the one in figure 3.

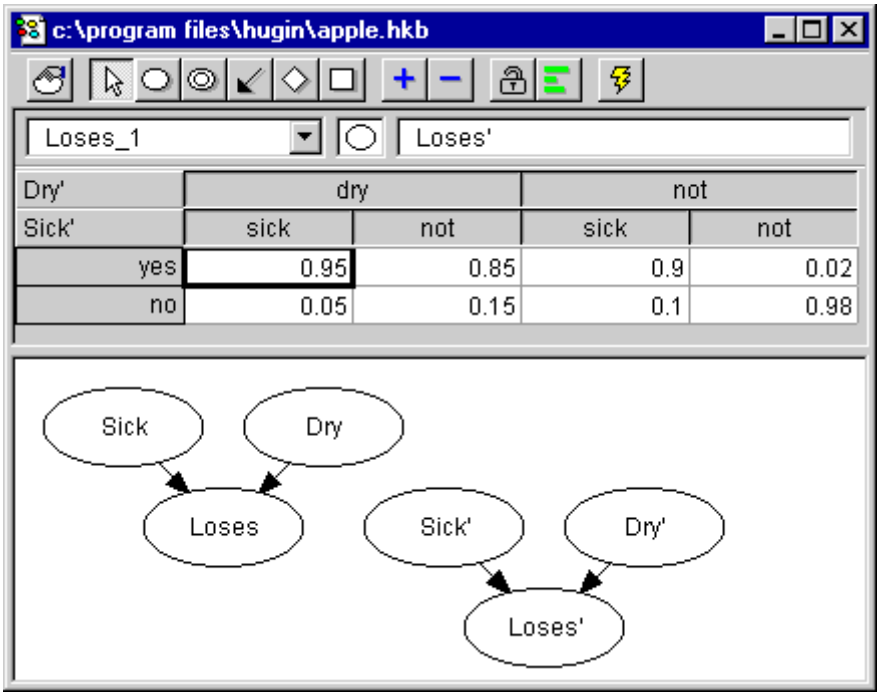

Figure 3: The BN extended with Sick', Dry' and Loses'

The next step is to add causal arrows from Sick to Sick' and from Dry to Dry'. If you do not remember how to do this, these steps will help you create one from Sick to Sick':

- Press the causal arrow tool
- Drag an arrow from Sick to Sick' with the left mouse button (while holding down the SHIFT key)

Holding down the SHIFT key enables you to create more causal arrows sequentially without having to press the causal arrow tool again and again.

Adding a Utility Node

So far, the network we have constructed is still a BN. Now, we shall make the first change making it an influence diagram. This change is the addition of a utility node. The utility node we shall add is the Harv node (see figure 1) representing the utility gained from the harvest. Here is how to add it:

- Press the utility tool (to the right of the causal arrow tool)
- Click somewhere in the network pane (a good place would be in the lower right corner besides the Loses' node
- Change the name and label of the new utility node to "Harv"

The harvest depends on the state of Sick' and thus there is an arrow from Sick' to Harv. Add this arrow:

- Press the causal arrow tool
- Drag an arrow from Sick' to Harv

The utility of the harvest was specified to that found in table 1.

Sick'="sick"	**Sick'**="not"
3000	20000

Table 1: U(Harv).

You enter the values of table 1 into the utility table of Harv as follows:

- Choose the Harv node as the currently active node by selecting it from the drop down list below the tool bar or simply by double clicking it
- Enter the values from table 1 in the utility table in the node edit pane

A Decision Node and one more Utility Node

Now, you are about to add the decision node Treat (see figure 1). This is done similar to the way you add chance nodes and utility nodes:

- Press the decision tool (to the right of the utility tool)
- Click somewhere in the network pane (a good place would be to the right of the Dry node)
- Change the name and label of the new decision node to "Treat"

You add an action to a decision node in the same way as you add a state to a chance node:

- Choose the Treat node as the currently active node by selecting it from the drop down list below the tool bar or simply by double clicking it
- Press the add state tool
- Change the action names to "treat" and "not"

The Treat decision node has an impact on the Sick' node so:

- Add an arrow from Treat to Sick'

The new decision node represents the decision to give the tree some treatment or not. If the plantation owner (Apple Jack) chooses to give treatment this will cost him something which shall be modeled by the Cost utility node. The Cost node has the utility table shown in table 2.

Treat="treat"	**Treat**="not"
-8000	0

Table 2: U(Cost).

Now, add the Cost utility node to the influence diagram:

- Add a new utility node (a good place would be to the right of the Treat node)
- Change the name and label of this node to "Cost"
- Add an arrow from Treat to Cost
- Fill in table 2 in the utility table of Cost

Filling in CPTs

When we copied the nodes Sick' and Dry', they inherited the CPTs of Sick and Dry. However, as both these nodes have become children of other nodes, their CPTs are no longer correct. Their new CPTs were specified to those found in table 3 and table 4.

- Fill in table 3 as the cpt of Sick'
- Fill in table 4 as the cpt of Dry'

	Treat="treat"		**Treat**="not"	
	Sick="sick"	**Sick**="not"	**Sick**="sick"	**Sick**="not"
Sick'="sick"	0.20	0.01	0.99	0.02
Sick'="not"	0.80	0.99	0.01	0.98

Table 3: P(Sick' | Sick, Treat).

	Dry="dry"	**Dry**="not"
Dry'="dry"	0.60	0.05
Dry'="not"	0.40	0.95

Table 4: P(Dry' | Dry).

Now your (limited memory) influence diagram (LIMID) is finished and it should look like the one in figure 4. At this point it would be a good idea to save your LIMID.

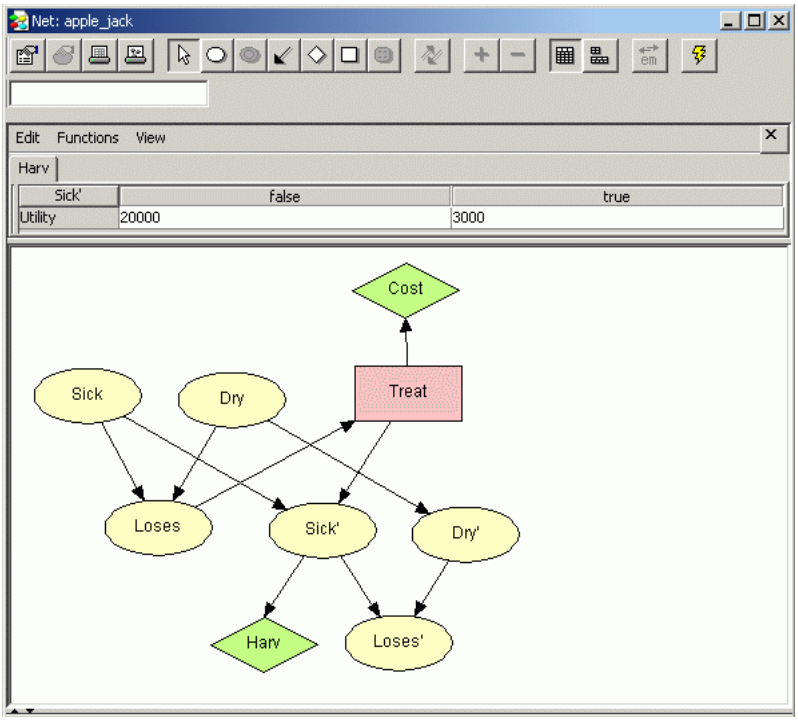

Figure 4: The complete influence diagram

Compiling the Limited Memory Influence Diagram

You can now try out the LIMID and hopefully you are eager to see how it works. First, compile the LIMID:

- Press the compile tool (the right most tool button in the network window tool bar)

The compilation of an influence diagram may produce some of the same errors as described in the first tutorial. If the LIMID does not compile, you have probably made some minor error. Once the influence diagram has been compiled, probabilities and expected utilities are computed under the initial policy. To solve the influence diagram, it is necessary to invoke Single Policy Updating.

What Should Apple Jack Do?

When the LIMID has been compiled, you should do a Single Policy Updating. Now, imagine that the only thing Jack knows about his tree is that it is losing leaves. Then, what will be the best thing for him to do? To find out this, follow these steps:

- Expand the Loses chance node and the Treat decision node in the node list pane on the left (by double clicking them)
- Enter the evidence that Loses is "yes" (by double clicking the "yes" state)
- Propagate the influence diagram (press the sum propagation tool)
- Read the expected utility of "treat" and "not" in the Treat decision node

You should be reading something looking like that in figure 5.

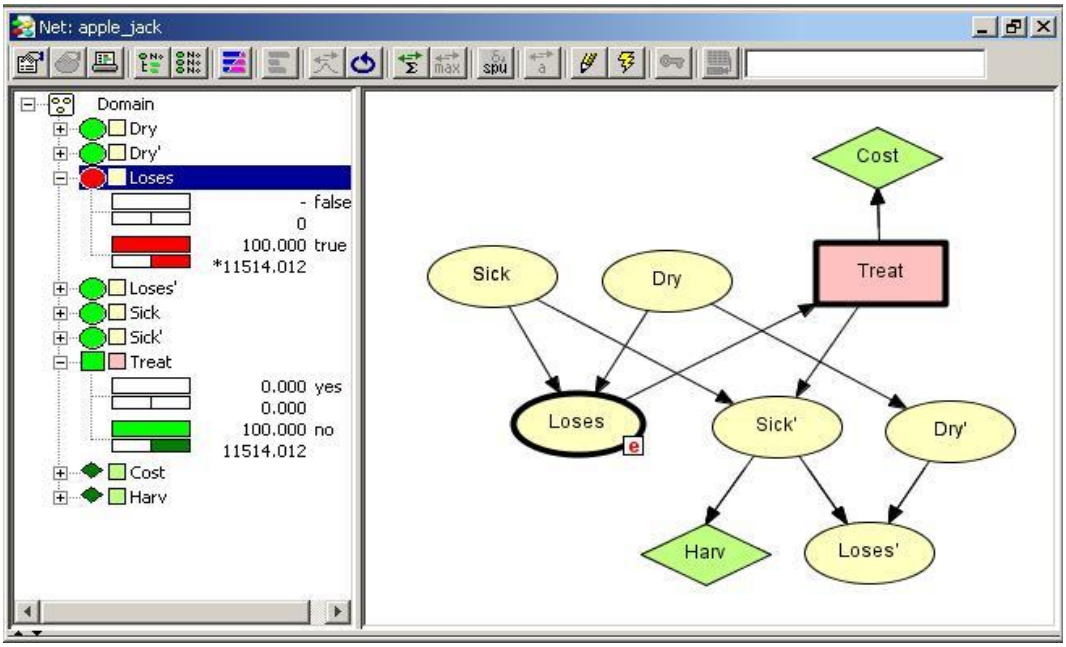

Figure 5: The influence diagram propagated with the evidence that Loses="yes".

You read 11514.0 as the expected utility of not doing anything. This suggests that it will be best for Apple Jack not to treat the tree.

This finishes the tutorial. You should now be able use the Hugin Graphical User Interface to construct your own (limited-memory) influence diagrams. However, creating large and complex models will require a more in-depth study into this subject area.

Tutorial 3. Structure Learning

Currently, the Hugin GUI supports two kinds of learning: structure learning and parameter learning. Structure learning is the process where the system learns the dependencies between the variables that exist in the data. Parameter learning is the task where you fill in the parameters describing the strength of the dependencies in the learned (or built) structure.

Structure Learning

Structure learning in Hugin is supported through the PC-algorithm. Consider the data file asia_no_e.dat for the ASIA_NO_E network. Figure 1 shows the first few lines of the data file.

```
X,B,D,A,S,L,T
no,no,no,no,no,no,no
yes,yes,yes,no,yes,yes,no
yes,yes,yes,no,yes,yes,no
no,no,yes,no,no,no,no
no,no,no,no,no,no,no
no,yes,yes,no,yes,no,no
no,yes,no,no,N/A,no,no
no,no,no,yes,yes,no,no
no,yes,yes,no,yes,no,no
no,no,no,no,no,no,no
no,yes,no,no,N/A,no,no
no,no,no,no,yes,no,no
no,no,no,no,no,no,no
no,yes,yes,no,yes,no,no
no,no,no,no,no,no,no
```

Figure 1: The data file

The structure learning functionality is available under the "File" menu and through the structure learning icon. The structure learning icon is shown in figure 2.

Figure 2: The structure learning window

Click the SL icon and the structure learning window appears. Notice the field "Significance level" which specifies the significance level of the statistical independence tests performed during structural learning. Press "Select File" and choose a file that will be used to estimate the structure. Then select the "OK" button as shown in figure 3.

Figure 3: The structural learning window

Pressing "OK" starts the structure learning algorithm. Based on the database of cases given in the file, the structure learning algorithm learns the structure of the graph of the Bayesian network. Figure 4 shows the result of structure learning based on the asia_no_e.dat file.

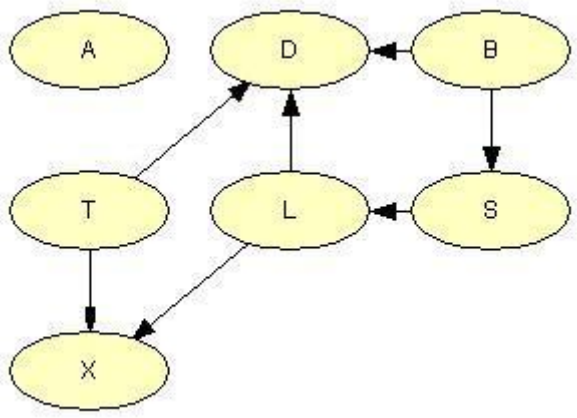

Figure 4: The learned Bayesian network graph

The network graph from which the data file has been sampled is shown in figure 5. When comparing the original network graph with the learned network graph, the only visible difference is the link from the "Visit to Asia"-node to the "Tuberculosis"-node is missing in the learned network graph. This is due to the fact that the strength of the dependency between these two nodes is rather weak. If the "Significance Level Of Dependency Test" factor were set to a higher value, this link would most likely be identified as well. However, other (incorrect) links may also be identified if the "Significance Level Of Dependency Test" factor is raised.

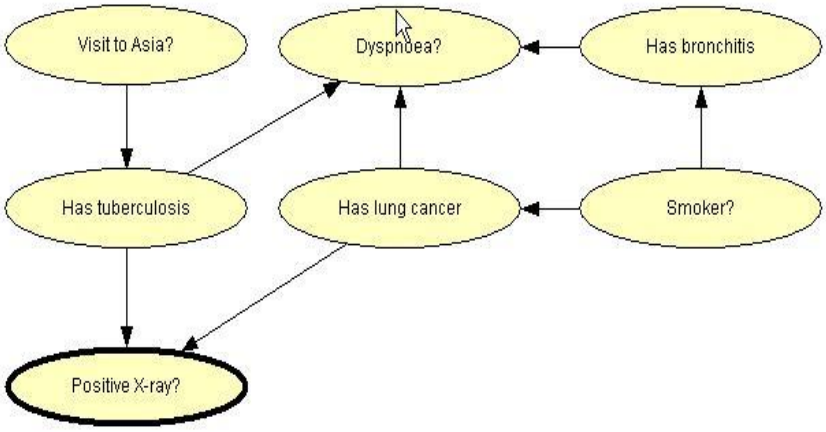

Figure 5: The Bayesian network graph used for sampling the data file

Once the structure of the Bayesian network graph is constructed, the conditional probability distributions of the Bayesian network can be estimated from data using the EM-learning algorithm, see the EM Learning Tutorial.

Tutorial 4. Parameter Learning / Adaptive Learning

The Hugin GUI supports two kinds of parametric learning; adaptive learning and EM (batch) learning.

Adaptive Learning

Adaptation (adaptive learning) is used to adapt all or some of your conditional probability tables in the knowledge base to a new set of data.

Consider the Bayesian network (BN) shown in figure 1. To learn more about the domain of this BN refer to the section samples.

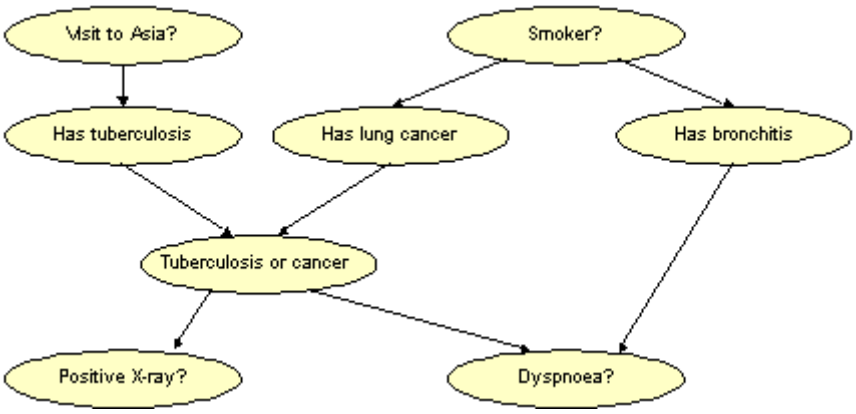

Figure 1: BN Representation of "Chest Clinic"

Experience table/ tables are necessary to use adaptation. To add an experience table to all discrete chance nodes in the domain, click somewhere in the network pane, right click and select "Add Experience Table to All Discrete Chance Nodes". It also possible to add experience tables to a subset of discrete chance nodes in the domain. To do the latter, select a specific discrete chance node then right click and select "Add Experience Table".

In our example it is justified to add experience tables to all the nodes in the domain except "Tuberculosis or cancer" since this node is a logical or, and no experience can be gained from logical or nodes. To see the created experience table, select the desired node and right click then select "Show Experience Table".

As mentioned earlier, the experience table of a node represents the experience counts of the parent configurations. For example, the experience table for node "Dyspnoea?" which is shown in figure 2, represents the number of observations of different parent configurations.

Has bronchitis	Yes		no	
Tuberculosis or cancer	Yes	no	yes	no
Experience count(s)	0	0	0	0

Figure 2: Experience Table for "Dyspnoea?" node

The value zero is an invalid experience count thus the value must be greater than 0 to activate adaptation. If our believe of the correctness of the present conditional distribution probability is high then the experience count must have a high value otherwise the value of the count should be low. In this case we assume that our believe in the correctness of the current conditional distribution is low thus we set the initial experience count to a low number, for instants "10". Figure 3 show the initial experience count table for "Dyspnoea?" discrete chance node.

Has bronchitis	yes		no	
Tuberculosis or cancer	Yes	no	Yes	No
Experience count(s)	10	10	10	10

Figure 3: Initial Experience Count Table for "Dyspnoea?" discrete chance node

Note that it is not necessary to activate experience count or enter the same experience count for every parent configuration. For instants the initial experience count values can be set to "10,0,100,0". Note that adaptation requires at least a node with experience table otherwise it is not possible to adapt the domain. Now add experience table to the every node in the BN (except "Tuberculosis or cancer" node) and set the initial experience counts to 10. The domain is now ready for adaptation.

An adaptation step consists of entering evidence, propagating, and finally updating (adapting) the conditional probability and experience tables.

Let's concentrate on one of the nodes namely "Smoker? The conditional distribution probability of this node prior to any adoption is S (0.5,0.5). Now enter the following observations:

- Node S is in state 0 ("yes")
- Node X is in state 0 ("yes")
- Node D is in state 0 ("yes")

then propagate the evidence. Next push the adaptation button which is shown in figure 4.

Figure 4: The Adaptation Icon

Keep clicking on the adaptation button a couple of times. Each time the adaptation button is pushed the probability of this observation (i.e. P(All)) increases. Now initialize the BN and observe the conditional distribution probability of the "Smoker? As you can see the conditional distribution probability is no longer S(0.5,0.5). Actually no conditional distribution probability is the same. This indicates that based on the new observations the conditional distribution probabilities has been changed. I.e. if the experience tables are now deleted or the values of the experience tables are set to zero then the current distribution probabilities will be the new conditional distribution probabilities of the nodes.

To extend the adaptation with optional fading we need to add fading table to some or all of the discrete chance nodes in the BN. To add fading tables to all discrete chance nodes in the BN click somewhere in the network pane, push the right mouse button and then choose *"Add Fading Table to All Discrete Chance Nodes"*. To add fading tables to a discrete chance node in the BN select the specific discrete chance node then right click and select *"Add Fading Table"*. Note that if a node does not have an experience table then it is not possible to add fading table to the node.

Tutorial 5. Parameter Learning / EM Learning

EM (Estimation-Maximization) is a learning facility used for batch learning. Batch learning is "off-line" learning and is used to generate conditional probability tables in the knowledge base from data stored in a database.

EM Learning

Consider the Bayesian network from the Tutorial 4. Adaptation. Figure 1 shows the current distribution probability for the net.

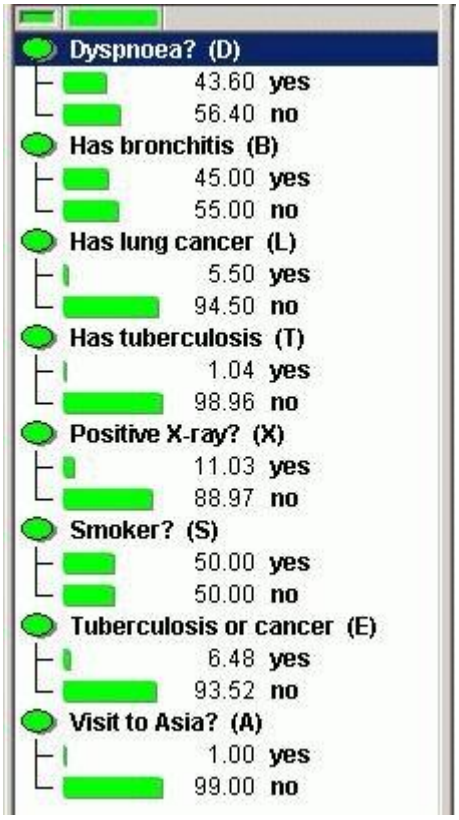

Figure 1: Current Distribution Probabilities

Now go back to "edit mode" and set all the conditional distribution probability to 1 except "Tuberculosis or cancer" node. Switch back to "run mode" and select the "EM-Learning" icon as shown in Figure 2.

Figure 2: The EM-Learning Icon

After clicking the icon, the EM learning window appears. Notice the field "Convergence threshold" response to the log-likelihood. Click "Select File" and choose a file from which conditional distribution probabilities will be computed. After selecting the file, the "OK" button appears as shown in Figure 3.

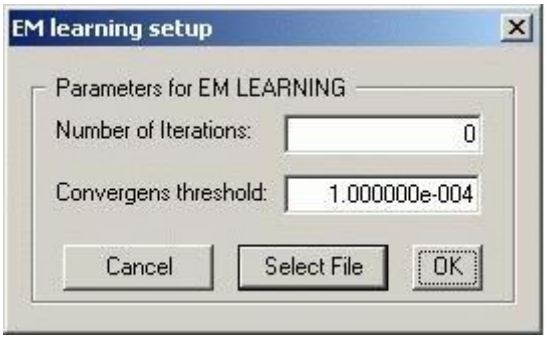

Figure 3: The EM-Learning Window

Clicking "OK" starts the EM-algorithm. Based on the case set given in the file, the EM-algorithm computes the conditional distribution probability for each node. Figure 4 shows the new conditional distribution probability based on asia.dat file.

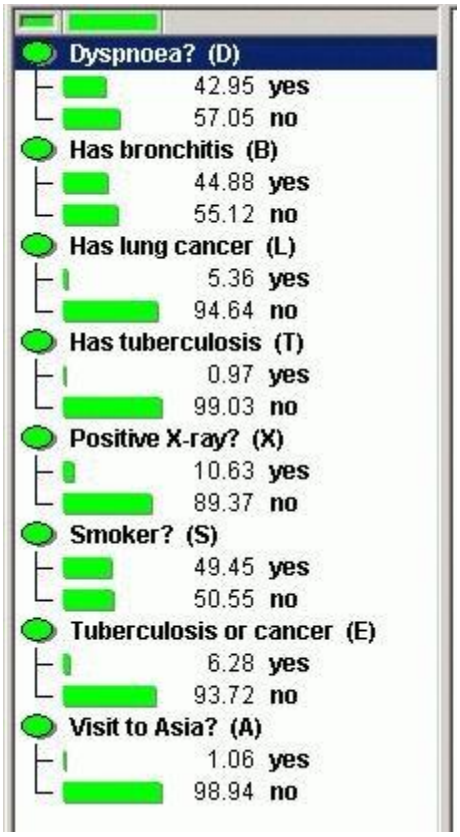

Figure 4: The New Conditional Distribution Probabilities

Tutorial 6. Working with Object-Oriented Bayesian Networks

By: Uffe Kjærulff

This tutorial shows how to implement a small object-oriented Bayesian network (OOBN) in HUGIN GUI. The OOBN we are about to construct is the one modelled in the Diseases example in the Basic Concepts section. Two other tutorials are available: In the second tutorial we show how to construct the Apple Tree example from the Basic Concepts section, and in the third tutorial we will extend this example to an influence diagram.

The qualitative (or structural) representation of our OOBN is shown in Figure 1. In this tutorial, we shall ignore the specification of the CPTs.

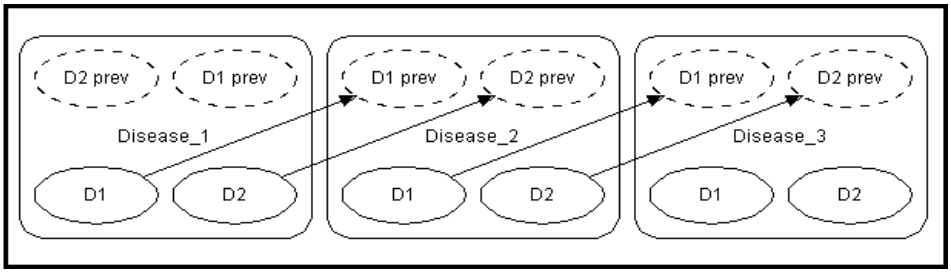

Figure 1: OOBN representing the Diseases problem

If you want to understand the design of this OOBN, you should read about it in the Basic Concepts section. Were we to construct this time-sliced network as a BN, we would get the network shown in Figure 2.

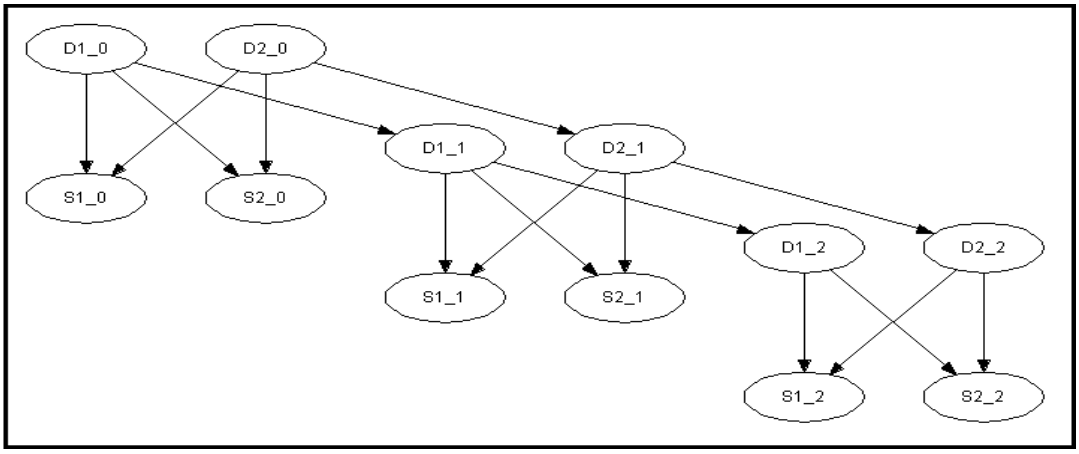

Figure 2: BN representation of the Diseases problem

Creating the Time-Slice Model

A BN is a special case of an OOBN. What makes a network an OOBN is the existence of instance nodes (i.e., nodes that represent instances of other networks). Thus, we start out by creating a new empty network by selecting the "New" menu item in the "File" menu. This gives us a new network window containing an empty network called "unnamed ", where x is some integer. It starts up in Edit Mode which allows us to start constructing the OOBN immediately (the other main mode is Run Mode which allows you to use the OOBN).

In Figure 2, we observe that each of the three time slices contains four nodes: D1, D2, S1, and S2, where D1 and D2 represent two different diseases with states "Present" and "Absent", and S1 and S2 represent symptoms that both may be observed as consequences of each of the diseases. We shall assume that S1 and S2 represent symptoms with two possible outcomes, "Observed" and "Unobserved". As each of the time slices are identical, both at the qualitative (or

structural) level and quantitative level (i.e., the CPTs are identical, including those that describe the temporal aspect, namely P(D1_2|D1_1), P(D2_2|D2_1), etc.), we need only construct a model describing a generic time slice and then connect three instances of this network.

Creating the Nodes

First, we construct the generic time-slice model, containing the four nodes D1, D2, S1, and S2. The nodes all represent discrete chance variables. Therefore, we select the Discrete Chance Tool and create the four nodes by clicking the left mouse button at four different locations in the network pane while keeping the Shift key down (to avoid reselecting the tool for each node). We then change the default names of the nodes and their default state names using the Node Properties pane. Second, we select the Link Tool and create the link from D1 to S1 by dragging the mouse cursor (i.e., pressing the left mouse button and moving the mouse cursor while keeping the button pressed) from a point inside D1 to a point inside S1 and then releasing the mouse button. Again, we keep the Shift key pressed, and create the other three links in the same manner. The result is illustrated in Figure 3.

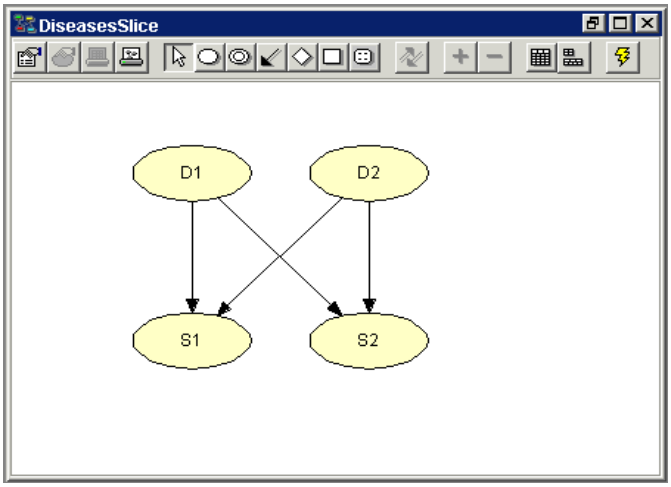

Figure 3: BN for a single time slice of the Diseases problem

Output Nodes

Now, in order for a network for a single time-slice have parent nodes in the immediately preceding network, we need to be able to refer to nodes outside the network in Figure 3. In a conventional BN, this is not possible. Thus, as D1 and D2 are going to be parents of D1 and D2, respectively, in the next time slice, we must declare D1 and D2 as output nodes, making them visible outside the network (or rather through instances of the network).

Input Nodes

Also, in the network in Figure 3, we should be able to specify the temporal aspect, namely the CPTs P(D1|D1 prev) and P(D2|D2 prev), where the nodes "D1 prev" and "D2 prev" are placeholder nodes for D1 and D2, respectively, in the immediately preceding time slice. Such placeholder nodes are referred to as input nodes, and shouldn't be confused with real nodes. A real node, which is type consistent with an input node, can be bound to that input node. That is, an input node becomes identical with the node that is bound to it. However, if an input node hasn't got a binding associated with it, the network containing the input node can still be used (i.e., compiled in to a junction tree and used for inference). In that case the input node is treated as a real node. That is, each input node has a CPT associated with it just as any ordinary node, but this CPT is used only if no nodes have been bound to the input node in a network containing an instance of the network in which the input node is defined.

Input nodes and output nodes are collectively referred to as interface nodes.

Creating the Interface Nodes

Now, let's try to put all this into practice. First, we declare D1 and D2 as output nodes. This is done by clicking the "Output" check box in the Node Properties pane for each of them. The color of the D1 and D2 then changes the the selected color for interface nodes (as set in the Network Properties pane) to indicate their new status as output nodes.

To create the two input nodes, "D1 prev" and "D2 prev", we first create two ordinary nodes and set their names to D1_prev and D2_prev (and/or their labels to "D1 prev" and "D2 prev"), respectively, in the Node Properties pane. Also, in the Node Properties pane for each of these two new nodes, we click the "Input" check boxes to declare them as input nodes. Similar to D1 and D2, the color of "D1 prev" and "D2 prev" changes the the selected color for interface nodes. In addition, the appearance of the borders of "D1 prev" and "D2 prev" changes from solid to dashed, which indicates that they are not real nodes.

Finally, we create links from "D1 prev" and "D2 prev" to D1 and D2, respectively. The result of these operations appear in Figure 4.

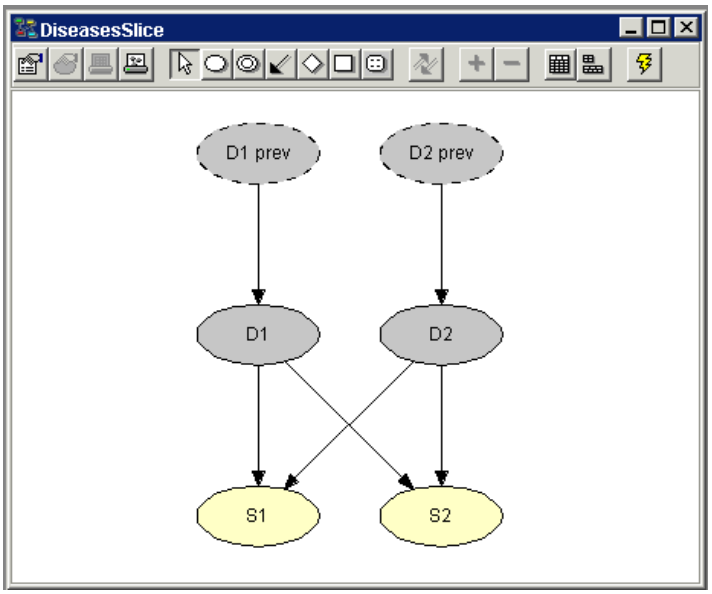

Figure 4: BN for a single time slice of the Diseases problem, including specifications of interface nodes

Creating the Diseases Model

To create the final Diseases model spanning three time slices, we first create a new empty network (via the "New" menu item in the "File" menu). Next, we select the Instance Tool and create three instance nodes by clicking the left mouse button at three different locations in the network pane while keeping the Shift key down (see Figure 5).

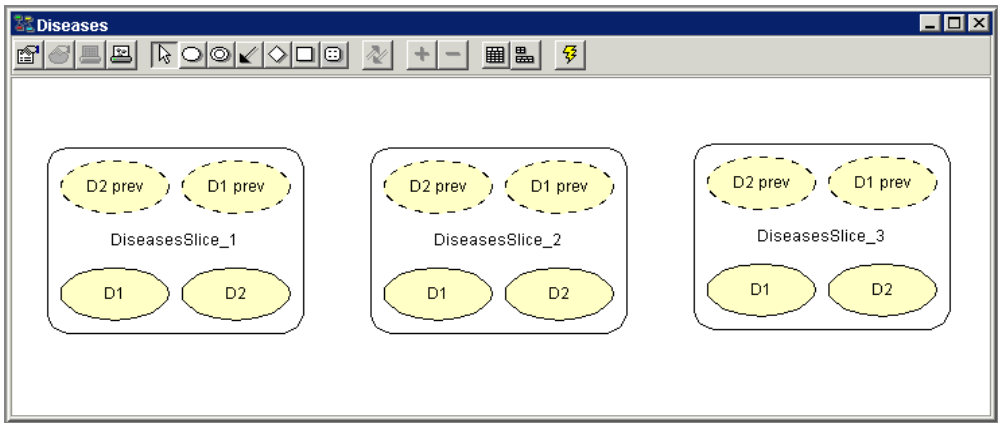

Figure 5: The network pane contains the three instance nodes that represent three instances of the time-slice model shown in Figure 4

Each instance node appears as a rectangle with rounded corners. We note that the nodes declared as interface nodes in the generic time-slice model appear in each of the instance nodes. The input nodes appear in a row at the top of the instance node, and, similarly, the output nodes appear at the bottom of the instance node.

Next, we need to bind the output nodes of instances DiseasesSlice_1 and DiseasesSlice_2 to the input nodes of instances DiseasesSlice_2 and DiseasesSlice_3, respectively. This is done by creating links (via the Link Tool) from the output nodes to the corresponding input nodes. The resulting model appears in Figure 6.

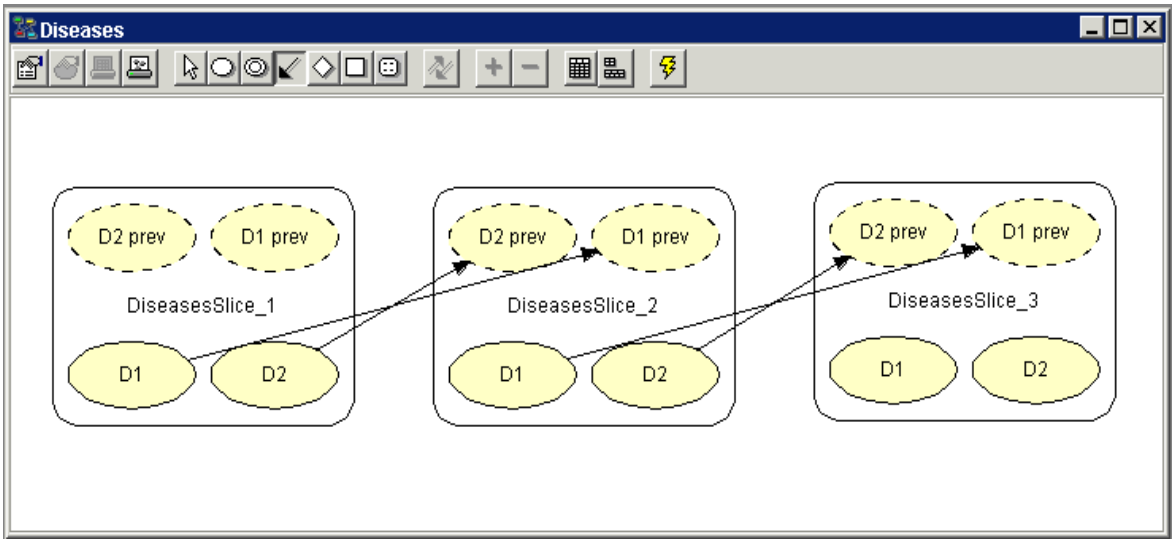

Figure 6: The output nodes of an instance are bound to input nodes of another instance using the Link Tool

Obviously, the model would look nicer if the order of appearance of the input nodes were reversed. If we select the Select Tool, we can easily alter the order of appearance of the interface nodes. We move an interface node one position to the left (right if the Shift key is down) by placing the mouse cursor on top of the interface node and clicking the left mouse button. After reordering the network appears as in Figure 7.

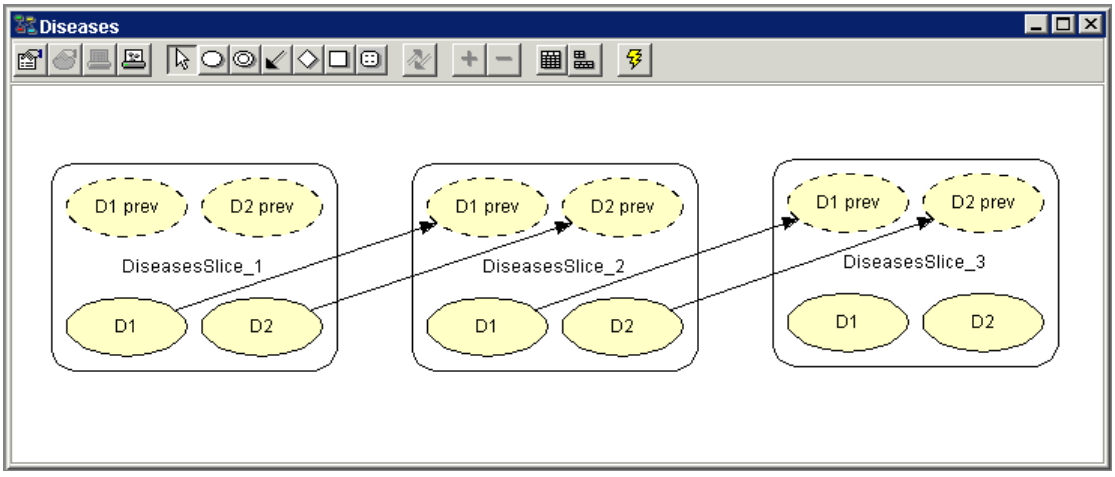

Figure 7: The order of appearance of interface nodes can be altered by simple mouse clicks

Finally, it is often useful to be able to collapse one or more of the interface nodes in order to hide irrelevant details, thereby making the network less cluttered. Again, if we activate the Select Tool, we can collapse (expand) an expanded (collapsed) instance node, simply by clicking the left mouse button just outside the node. Alternatively, we can choose the "Collapse Instance Nodes" ("Expand Instance Nodes") menu item in the "View" menu, which collapses (expands) all instance nodes.

The OOBN in Figure 7 with the instance nodes collapsed is shown in Figure 8.

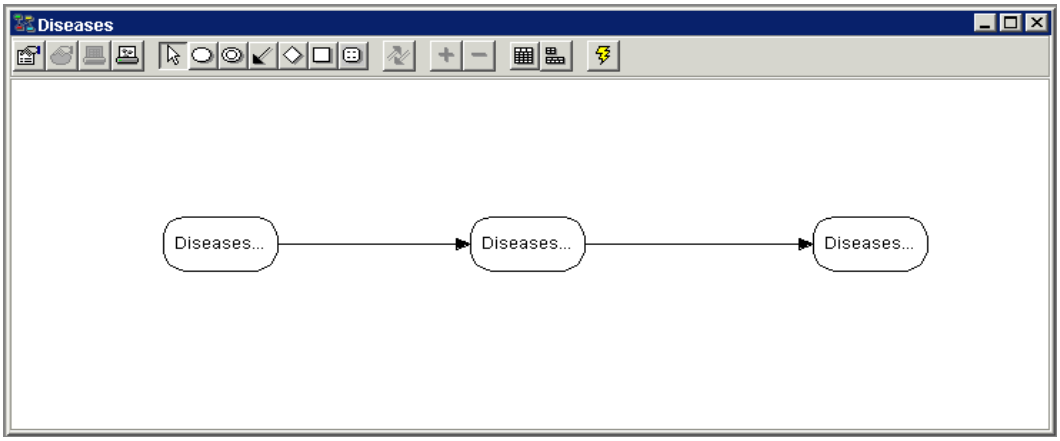

Figure 8: The OOBN in Figure 7 with the instance nodes collapsed

Compiling the OOBN

Now, assuming the CPTs of the time-slice model in Figure 4 has been filled in (see the tutorial on BNs for details), it is the time to compile the network and see how it works:

- Press the Run Mode tool button in the tool bar (see Figure 9)

Figure 9: The Run Mode tool button

- For each configuration of parent states in the CPT of a node the probabilities of the different states of the node must sum to 1. In other words, each column of the table must sum to 1. If there is a column that does not sum to 1, the compiler will normalize the values. This fact can be utilized when filling in the probabilities. Say, for example, that the probability of D1=Present in the first time slice is based on the observation of 13527 patients, 168 of whom were observed to have the disease. Instead of first calculating the fractions, you just put 168 in the Present state of D1, and 13359 in the Absent state. Then the compiler will calculate the proper values.

Running the OOBN

In Run Mode, the network window is split into two by a vertical bar (see Figure 10). To the left is the node list pane and to the right is the network pane.

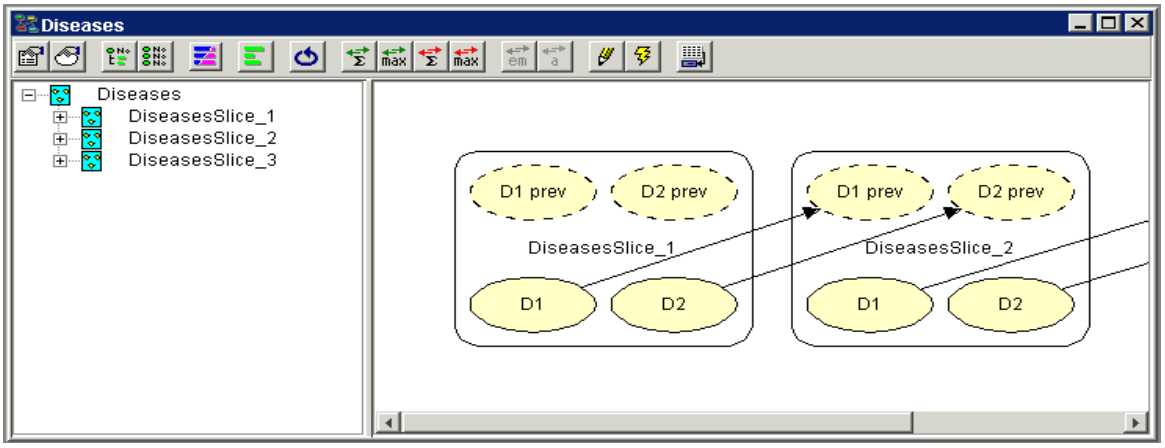

Figure 10: The network window in Run Mode.To the left is the node list pane (with all nodes collapsed) and to the right is the network pane

You can view the probabilities of a node being in a certain state by expanding the node in the node list pane. You expand (collapse) a node by clicking its plus (minus) icon in the node list pane. A node can also be expanded (collapsed) by selecting (deselecting) it in the network pane. You can also expand (collapse) all nodes at once by pressing the expand (collapse) node list tool in the tool bar just to the right of the node properties tool.

Unlike basic nodes, instance nodes don't have belief monitors associated with them, as they represent entire (sub)networks. Instead we must expand the instance node, whereby we get to see the list of nodes of the (sub)network that the instance node represents (see Figure 11).

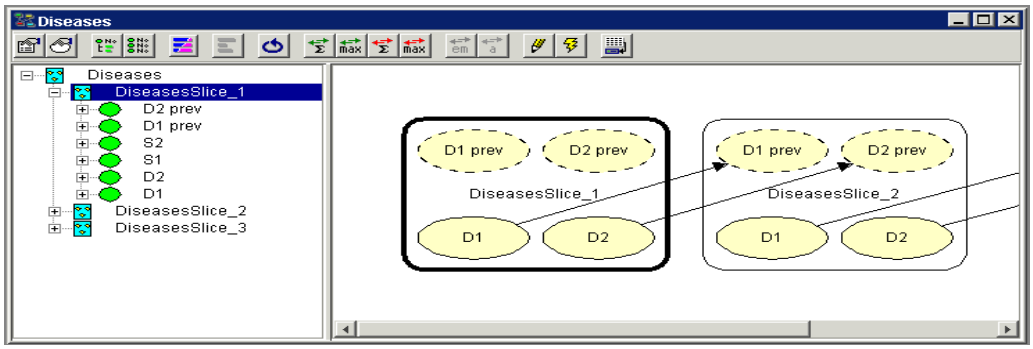

Figure 11: The nodes of the network represented by an instance node, get displayed in the node list pane when the instance node gets selected

For more details on the Run Mode, please see Tutorial 1.

Tutorial 7. Using the Table Generator

This tutorial shows you how the table generator functionality can be used to simplify how tables are specified for discrete chance nodes.

Imagine you are playing a game of dice where you roll a number of dice ranging from 1 to 5. The more sixes you roll the better, so you are very interested in predicting how many sixes you can expect to roll.

A very simple Bayesian network can model your situation. Figure 1 shows a Bayesian network where the number of the dice rolled (nDice) has an impact on the number of sixes rolled (nSixes).

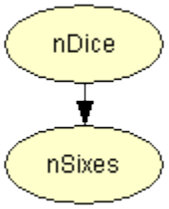

Figure 1: BN modelling the dice problem

nDice has states: 1, 2, 3, 4, 5 (the number of dice you could be asked to roll). nSixes has states: 0, 1, 2, 3, 4, 5 (one more than nDice since it is possible not to roll any sixes).

Now, because we want the states of the nodes to represent numeric values, we change their type to "Numbered". Explained below is how to change nDice to a numbered node:

- Select nDice with the mouse cursor
- Select the "Node Properties" item of the "Edit" menu
- Select the "Node" tab (if it is not already selected)
- From the "Type" drop down box select "Numbered"
- Click "OK"

Now, change nSixes to be numbered, too!

You can edit the number of states and the state values in the left most column of the node edit pane (as you probably did in Tutorial 1. However, you can also do it through the Node Properties dialog:

- Select nDice with the cursor
- Select "Node Properties" from the "Edit" menu
- Select the "States" tab
- Click the first state (the only state if you haven't added any states) in the states list
- Start typing "1" - this should transfer focus to the edit box below the states list
- Click the "Rename" button in the button list on the right
- Click in the edit box and type "2"
- Click the "Add After" button
- Continue entering the state values (now pressing the enter key after typing a state value should automatically use the "Add After" button since this was the last one activated in the button list)
- After typing all states, press "OK"

Now do the same for nSixes (remember to start with state "0").

Figure 2 shows the Node Properties dialog after entering the state values of nDice.

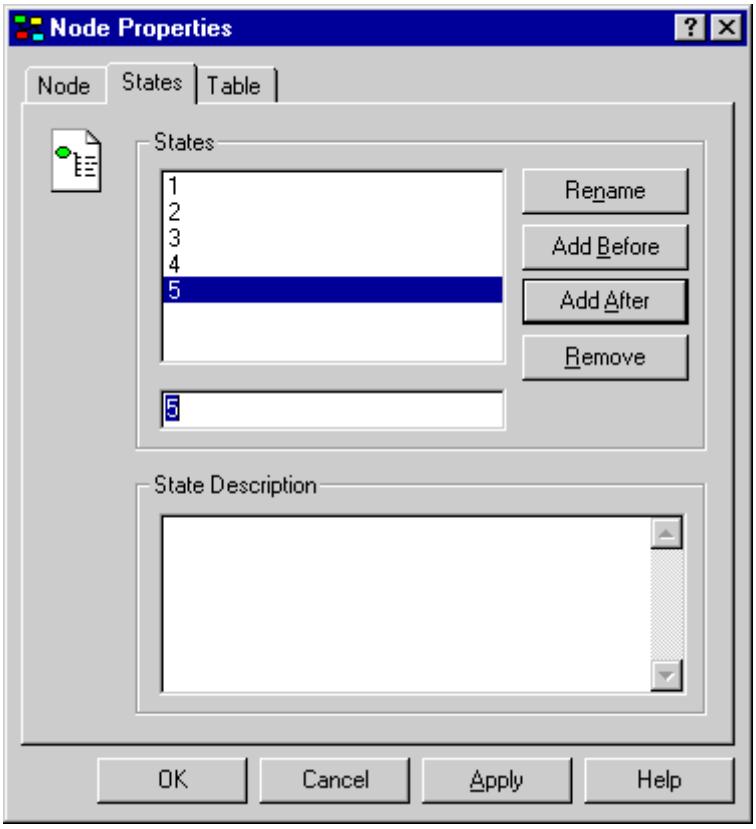

Figure 2: Entering the state values of nDice

The conditional probability table of nSixes in the node edit pane should now look like the one shown in figure 3.

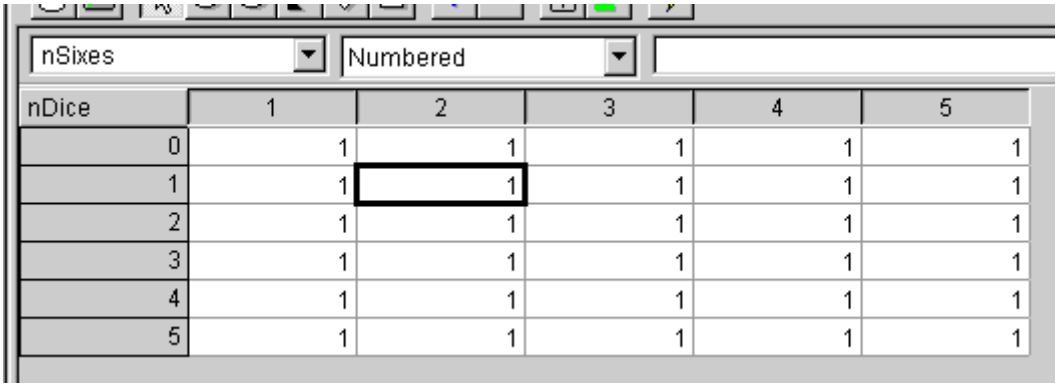

Figure 3: The conditional probability table of nSixes

This table is rather big and if you had to fill it in yourself it would be a pretty hard task. You would also have a tool for calculating the Binomial distribution, since the probability of rolling a specific number is a Binomial distribution depending on the number of dice you roll.

However, the Hugin GUI allows you to generate the table automatically from an expression you specify. To do this for the nSixes node follow these steps:

- Select nSixes as the currently selected node (so that it appears in the node edit pane)
- From the "Table" menu select "Expressions" (notice, that the look of the table changes dramatically - only one cell is now shown)
- Click in the single field of the table of nSixes
- From the "Table" menu select "Build Expression". This should open the "Expression Builder" dialog

- In the "Function Category" list select "Discrete Distributions" and in the "Function Name" list choose "Binomial", then press "OK". This should close the first dialog of the expression builder and open another one prompting you for the arguments to the Binomial distribution
- Click inside the edit box of the "n" argument
- Select nDice in the "Parents" list in the bottom and press "Insert"
- In the edit box of the "p" argument type "1/6"
- Press "OK"

Now, you should have the expression "Binomial (nDice, 1/6)" in the single field of the nSixes table. This is shown in Figure 4.

Figure 4: The expression table of nSixes after specifying a Binomial distribution depending on nDice

This ends the tutorial about building a small Bayesian network using the table generator. Try compiling it and play with it (try selecting different values for the nDice node and propagate). After compiling the network, you can also go back to edit mode and take a look at the generated table: Select the nSixes node and select "Manual" from the "Table" menu. You will be warned that you are destroying the expression, but that does not matter now. This can be rebuilt.

Tutorial 8. Case Generator

To generate cases based on the current conditional probability distribution, click the simulation icon as shown in Figure 1.

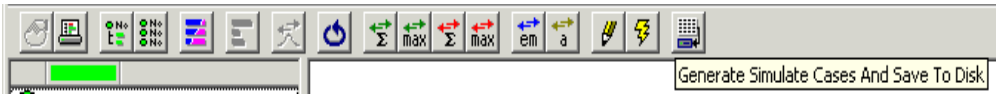

Figure 1: The Simulation Icon

By default the Hugin GUI chooses to create 10,000 cases with 5 percent missing values. The user can change these values. To generate the cases, one of the algorithms: MCAR or MAR must be selected. Figure 2 shows the case generator window.

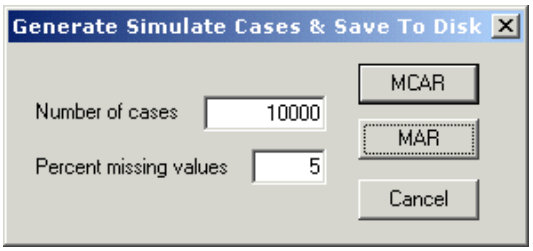

Figure 2: The Case Generator Window

Now the Hugin GUI will generate a set of cases and store these in a plain text file. Figure 3 shows the content of such a file where the cases are generated based on the Chest Clinic BN.

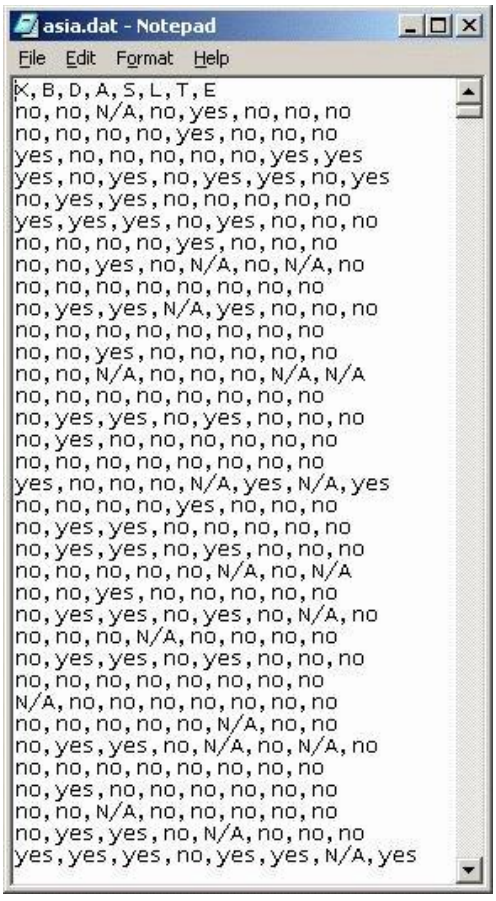

Figure 3: The Content of a Data File