HUGIN - The Tool for Bayesian Networks and Influence Diagrams

Frank Jensen, Uffe B. Kjærulff, Michael Lang, Anders L. Madsen
Hugin Expert A/S
Aalborg, Denmark

Abstract

In this paper, we describe the main functionality of the Hugin Tools — Hugin Decision Engine and Hugin Graphical User Interface — for using Bayesian networks and influence diagrams.

1 Introduction

Bayesian Networks (Pearl, 1988; Cowell et al., 1999) and influence diagrams (Howard and Matheson, 1981) are increasingly popular paradigms for reasoning and decision making under uncertainty, respectively.

The Hugin Tools support manual construction, learning, inference, and analysis of Bayesian networks and influence diagrams.

The Hugin Tools consist of the Hugin Decision Engine and the Hugin Graphical User Interface. A set of Application Programming Interfaces (APIs) give the user access to the functionality of the Hugin Decision Engine. APIs are available for the major programming languages C, C++, and Java. An ActiveX-server component is available for Visual Basic for Applications. Most of the functionality of the Hugin Decision Engine is available both in the APIs and in the Hugin Graphical User Interface.

In this paper we describe the main functionalities of the Hugin Decision Engine and the Hugin Graphical User Interface.

2 Hugin Decision Engine

The Hugin Decision Engine is the inference engine – the heart of the Hugin Tools and provides fundamental functionalities for model construction, learning, and analysis.

2.1 Knowledge Representation

The Hugin Tools provide a range of paradigms and methods for representation of knowledge in structured domains with inherent uncertainty. These tools are particularly useful when the domain knowledge include knowledge about structural relationships, which encode dependence and independence assumptions among quantities in the problem domain. These relationships, encoding the *qualitative domain knowledge*, can describe causal (directed) interactions or "symmetric" (undirected) interactions, and they can be derived from domain experts' experience, from data, or from both of these sources.

The *quantitative domain knowledge* is provided through local conditional probability distributions and utility functions.

The range of knowledge representation paradigms provided in the Hugin Tools include Bayesian networks (with purely discrete variables, purely continuous variables, or a mix of discrete and continuous variables), influence diagrams, and object-oriented Bayesian networks and influence diagrams. Also, powerful methods are available for specifying the quantitative knowledge and for (semi-)automatic learning of the qualitative as well as the quantitative knowledge hidden in a database of cases.

The structural relations of a Bayesian network or an influence diagram are encoded in a directed acyclic graph (DAG), $\mathsf{G} = (\mathsf{V}, \mathsf{E})$, over the set of domain quantities or variables, V, where each variable, $\mathsf{X} \in \mathsf{V}$, can have zero or more parent variables, $\pi(\mathsf{X})$ (i.e., variables connected to X through a directed link, $e \in \mathsf{E}$, emanating from each of these variables).

The dependence and independence relations among the variables depend on the observations available (i.e., observations of actual states

J.A. Gámez and A. Salmerón (Eds.): PGM 2002, pp. 212–221, 2002
Procs. of the 1st European Workshop on Probabilistic Graphical Models

or values of the variables). Simple rules are available for reading these relations from a DAG (Pearl, 1988; Lauritzen et al., 1990).

The strength of the (causal) relationships among a set of parent variables, $\pi(X)$, and a child variable, X, is encoded via a conditional probability distribution $P(X \mid \pi(X))$.

2.1.1 Bayesian Networks

A Bayesian network $N = (G, \mathcal{P})$ consists of a DAG, $G = (V, E)$, and a set of probability distributions $\mathcal{P}$. Each node $X \in V$ represents a unique random variable. (We use the terms "node" and "variable" interchangeably.) For each variable $X \in V$ there is a conditional probability distribution $P(X \mid \pi(X)) \in \mathcal{P}$.

In a Bayesian network of purely discrete variables, the joint probability distribution, $P(V)$, factorizes according to the structure of its DAG:

$$P(V) = \prod_{X \in V} P(X \mid \pi(X)).$$

This is the chain rule of Bayesian networks.

A famous toy example of a Bayesian network is the Chest Clinic example (Lauritzen and Spiegelhalter, 1988).

Example 2.1 [Chest Clinic]
Shortness-of-breath (dyspnoea) may be due to tuberculosis, lung cancer or bronchitis, or none of them, or more than one of them. A recent visit to Asia increases the chances of tuberculosis, while smoking is known to be a risk factor for both lung cancer and bronchitis. The result of a single chest X-ray does not discriminate between lung cancer and tuberculosis, as neither does the presence or absence of dyspnoea.

The qualitative knowledge of this diagnostic problem can be captured by the Bayesian network shown in Figure 1.

We leave out the specification of quantitative knowledge in all examples. □

2.1.2 Mixed Bayesian Networks

A mixed Bayesian network contains a mixture of discrete and continuous variables (Lauritzen, 1992; Lauritzen and Jensen, 2001). The discrete variables are denoted as V_Δ and the continuous ones as V_Γ.

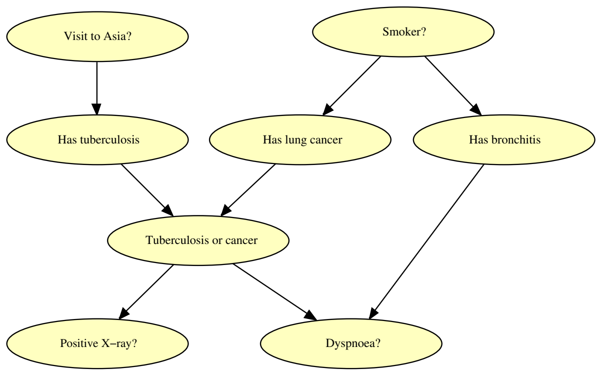

Figure 1: The Chest Clinic example.

The chain rule of mixed Bayesian networks is:

$$\prod_{X \in V_\Delta} P(X = i_X \mid \pi(X) = i_{\pi(X)}) * \mathbb{N}_{|V_\Gamma|}(\mu(i), \Sigma(i)),$$

for each configuration i of V_Δ.

The Waste Incinerator example (Lauritzen, 1992) illustrates the use of a mixed Bayesian network.

Example 2.2 [Waste Incinerator]
The emission from a waste incinerator differs because of compositional differences in incoming waste. Another important factor is the waste burning regime, which can be monitored by measuring the concentration of CO_2 in the emission. The filter efficiency depends on the technical state of the electro filter and the amount and composition of the waste. The emission of metal depends on both the concentration of metals in the incoming waste and the emission of dust particulates in general. The emission of dust is monitored through measuring the penetrability of light.

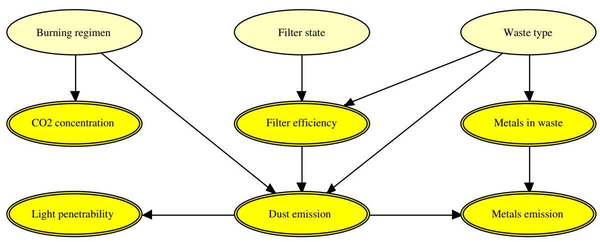

Figure 2: The Waste Incinerator example.

The qualitative knowledge of this example can be captured by the mixed Bayesian network shown in Figure 2. □

2.1.3 Influence Diagrams

An influence diagram $N = (G, \mathcal{P}, \mathcal{U})$ is a Bayesian network augmented with decision variables and utility functions. An influence diagram supports the representation and solution of sequential decision problems with multiple utility nodes under the no-forgetting assumption (i.e., perfect recall is assumed of all observations and decisions made in the past that are influential in a given decision situation).

The chain rule of influence diagrams is:

$$EU(V) = \prod_{X \in V_C} P(X \mid \pi(X)) \sum_{U \in V_U} u(\pi(U)).$$

The Oil Wildcatter example (Raiffa, 1968) illustrates the use of an influence diagram.

Example 2.3 [Oil Wildcatter]
An oil wildcatter must decide either to drill or not to drill. He is uncertain whether the hole is dry, wet, or soaking, The wildcatter could take seismic soundings that will help determine the geological structure of the site. The soundings will give a closed reflection pattern (indication of much oil), an open pattern (indication of some oil), or a diffuse pattern (almost no hope of oil).

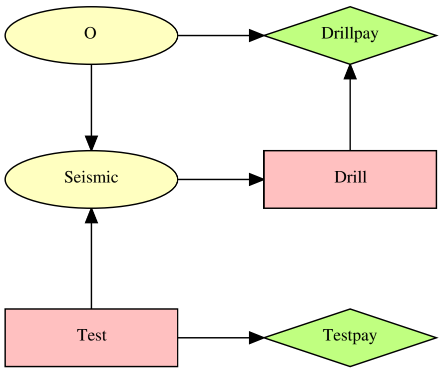

Figure 3: The Oil Wildcatter example.

The qualitative knowledge of this example can be captured by the influence diagram shown in Figure 3. □

2.1.4 Object-Oriented Modeling

The Hugin Tools support object-oriented specification of Bayesian networks and influence diagrams, which make it simple to reuse models, to encapsulate submodels (providing a means for hierarchical model specification), and to perform model construction in a top-down fashion, a bottom-up fashion, or a mix of the two (allowing repeated changes of level of abstraction).

In addition to the usual nodes, an object-oriented Bayesian network or influence diagram contains instance nodes, representing instances of other networks. In other words, an instance node represents a subnetwork. Of course, instance nodes can be nested, whereby an object-oriented network can be viewed as a hierarchical description of a problem domain. Describing a network in a hierarchical fashion often makes the network much less cluttered, and thus provides a much better means of communicating ideas among knowledge engineers and users.

As systems often are composed of collections of identical or similar components, models of systems often contain repetitive patterns. The notion of instance nodes makes it simple to construct multiple identical instances of a network fragment.

Example 2.4 [Stud Farm]
Consider a studfarm with 10 horses: Cecily has unknown mare and sire, John has mare Irene and sire Henry, Henry has mare Dorothy and sire Fred, Irene has mare Gwenn and sire Eric, Gween has mare Ann and unknown sire, Eric has mare Cecily and sire Brian, Fred has mare Ann and unknown sire, Brian has unknown mare and sire, Dorothy has mare Ann and sire Brian, and Ann has unknown mare and sire.

Given that John is a sick horse, the task is to compute the probability of each other horse being sick or a carrier.

The qualitative knowledge of this example can be captured by the object-oriented Bayesian network shown in Figure 4 where each node is a collapsed instance node. □

2.1.5 Table Generator

The specification of a conditional probability table $P(X \mid \pi(X))$ or a utility table $u(\pi(U))$ can

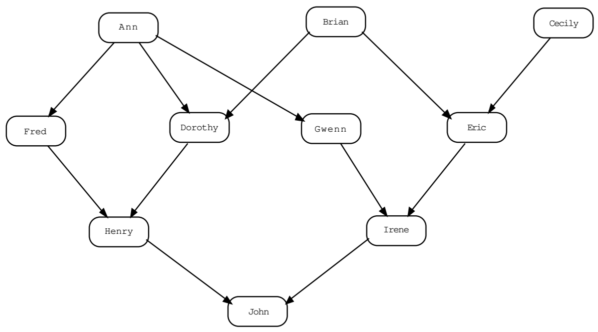

Figure 4: The Stud Farm example.

be a labour intensive task when the size of the parent set is large.

In many cases, however, $P(X \mid \pi(X))$ or $u(\pi(U))$ are known to follow (at least approximately) certain functional or distributional forms. The Hugin Tools provide a powerful Table Generator mechanism for compact specifications of such conditional probability and utility tables. The Table Generator provides a large set of standard mathematical functions for specifying table models and expressions.

An expression is built using standard statistical distributions (e.g., Normal, Binomial, Beta, Gamma, etc.), arithmetic operators, standard mathematical functions (e.g., logarithmic, exponential, trigonometric, and hyperbolic functions), logical operators (e.g., and, or, if-then-else), and relations (e.g., less-than, equals).

The different operators used in an expression have different return types and different type requirements for arguments. Thus, in order to provide a rich language for specifying expressions, it is convenient to have a classification of the discrete chance and decision nodes into different groups. The Hugin Decision Engine accepts four different types of discrete variables: *labeled, Boolean, numbered, and interval* variables.

Example 2.5 [Fair or Fake Die ?]
Consider the problem of computing the probabilities of getting n 6's in n rolls with a fair die or a fake die, respectively. A random variable, X, denoting the number of 6's obtained in n rolls with a fair die is Binomially distributed with parameters $(n, 1/6)$. Thus, the probability of getting k 6's in n rolls with a fair die is $P(X = k)$, where P is Binomial$(n, 1/6)$. Assuming that for a fake die the probability of getting 6 eyes in one roll is $1/5$, the probability of getting k 6's in n rolls with a fake die is $Q(X = k)$, where Q is Binomial$(n, 1/5)$.

A Bayesian-network model of this problem contains a node "n6s" with parents "nRolls" and "fakeDie". Now, if we let "nRolls" be a numbered node with states $1, \ldots, n$, let "fakeDie" be a Boolean node, and let "n6s" be a numbered node with states $0, \ldots, n$, then $P(\text{n6s} \mid \text{nRolls}, \text{fakeDie})$ can be specified elegantly through the expression:

Binomial(nRolls, if (fakeDie, 1/5, 1/6)).

□

Example 2.6 [Discretization]
Assume that $P(X|Y)$ can be approximated by a Normal distribution with mean given by Y and with variance 1, where Y is a numbered or an interval variable. Now, a suitable discretization of X is determined (e.g., given by the intervals $(-\infty, -5)$, $[-5, -2)$, $[-2, 0)$, $[0, 2)$, $[2, 5)$, $[5, \infty)$) and an interval variable, say X', is defined with states corresponding to the discretization intervals. Then we can specify $P(X' \mid Y)$ simply as Normal$(Y, 1)$. □

2.2 Structural Learning, Parameter Estimation, and Adaptation

Three different kinds of learning are included in the tools: structural learning, parameter estimation, and sequential learning of parameters.

Structural learning of a Bayesian network amounts to deriving a DAG G from a database of cases $D = \{c_1, \ldots, c_N\}$. For efficiency reasons, we use the PC algorithm (Spirtes et al., 2000), which is a constraint-based algorithm. The main steps of the PC algorithm are:

1. Test for (conditional) dependence relations between each pair of variables $(X \perp Y \mid S)$.

2. Identify the skeleton of the graph.

3. Identify colliders $(X \rightarrow Y \leftarrow Z)$.

4. Identify derived directions.

Tests are performed using conditioning sets of size $0, 1, 2, 3$. If $X \perp Y \mid S$ is found to be satisfied, the testing for independence between X and Y is terminated. The identification of derived directions is based on a set of three simple rules (Meek, 1995), see Figure 5. The algorithm produces a partially directed acyclic graph (PDAG).

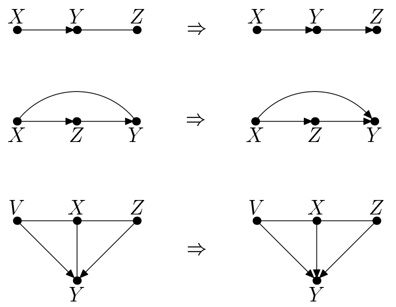

Figure 5: Three rules for orientation of edges.

The PC algorithm is extended with a Necessary Path Condition (NPC) (Steck, 2001). Informally, the NPC says that in order for two variables X and Y to be independent (in a DAG-faithful data set) conditional on a set $\mathcal{S}$ and no subset $\mathcal{S}' \subset \mathcal{S}$, there must exist a path between X and every $C \in \mathcal{S}$ (not crossing Y) and between Y and every $C \in \mathcal{S}$ (not crossing X). Otherwise, the inclusion of each C in $\mathcal{S}$ is unexplained. Thus, in order for an independence statement to be valid, a number of edges are required to be present in the graph. A maximal set of interdependent edges is called an ambiguous region. The NPC is a necessary condition for the existence of a perfect map given a set of conditional dependence and independence statements $\mathcal{M}$. The ambiguous regions indicate structural uncertainty. The user is intended to make decisions on the resolution of the inconsistencies in $\mathcal{M}$ and on the direction of edges when these cannot be determined from $\mathcal{M}$ alone. The NPC algorithm produces a PDAG with ambiguous regions.

The user is given the option to specify structural constraints on the presence and direction of edges. In the case of user specified constraints a fourth rule is needed to identify derived directions, see Figure 6, where a directed or undirected edge between X and Y is present.

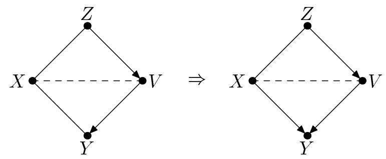

Figure 6: Fourth rule for orientation of edges.

Let $M = (G, \mathcal{P})$ be a Bayesian network. The entries of the conditional probability distribution are the parameters of M such that $\theta_{ijk} = P(X_i = k \mid \pi(X_i) = j)$ for each i, j, k. Parameter estimation is supported through the EM algorithm (Lauritzen, 1995).

The log-likelihood function $l(\theta)$ of the parameters θ given the data D and DAG G is:

$$l(\theta) = \sum_{i=1}^{N} \log P(c_i \mid \theta).$$

The E-step of the EM algorithm computes the current expected counts for family and parent configurations of each node X_i under θ:

$$n^*(Y) = \mathbb{E}_\theta\{n(Y) \mid D\},$$

where Y is either $\pi(X_i) = j$ or $X_i = k, \pi(X_i) = j$.

The M-step computes new estimates of θ^*_{ijk} given the current parameters θ_{ijk}:

$$\theta^*_{ijk} = \frac{n^*(X_i = k, \pi(X_i) = j)}{n^*(\pi(X_i) = j)}.$$

The above procedure is iterated until convergence of $l(\theta)$. When the difference between the log-likelihoods of two successive iterations is less than or equal to the numerical value of a threshold times the log-likelihood, the procedure is stopped. Alternatively, the user can specify an upper limit on the number of iterations to ensure that the procedure terminates.

Domain expert knowledge is specified as a Dirichlet distribution using the distribution $P(X_i \mid \pi(X_i)) = \{p_{ijk}\}$ and the experience counts $\alpha_{i1}, \ldots, \alpha_{i|\pi(X_i)|}$ associated with each node X_i. The M-step is then:

$$\theta^*_{ijk} = \frac{n^*(X_i = k, \pi(X) = j) + p_{ijk}\alpha_{ij}}{n^*(\pi(X) = j) + \alpha_{ij}}.$$

Using $l(\theta)$ as a quality measure does not consider the complexity of the model. In the future, different quality measures like the BIC score will be included.

Adaptation (Spiegelhalter and Lauritzen, 1990; Olesen et al., 1992; Cowell et al., 1999) is the task of sequentially estimating the parameters of the model as cases become available. Let an n-dimensional Dirichlet distribution $\mathcal{D}(\alpha_{i1}, \ldots, \alpha_{in})$ specify our prior belief in the parameter vector $\vec{\theta}_{ij}$. This is represented using an experience count α_{ij} and vector $\vec{p}_{ij}$. Thus, the experience count for a particular state k of X_i given $\pi(X_i) = j$ is $\alpha_{ijk} = \alpha_{ij} p_{ijk}$. After a complete observation on $(X_i = k, \pi(X_i) = j)$, the posterior belief is updated as $\alpha^*_{ijk} = \alpha_{ijk} + 1$ and $\alpha^*_{ijl} = \alpha_{ijl}$ for $l \neq k$. After an incomplete observation, the posterior over $\vec{\theta}_{ij}$ is approximated by a Dirichlet distribution with correct mean and variance for each parameter in order to avoid combinatorial explosion:

$$
\begin{aligned}
m^*_{ijk} &= \frac{\alpha_{ijk} + p_{ijk} + m_{ijk}(1 - p_{ijk})}{\alpha_{ij} + 1}, \\
v^*_{ijk} &= \frac{m_{ijk}(1 - m_{ijk})}{\alpha_{ij} + 1}.
\end{aligned}
$$

The updated experience count is computed from the mean and variance.

Before the adaptation takes place, the experience count α_{ij} is faded by a factor of λ_{ij} according to $p_{ij} = P(\pi(X_i) = j)$ such that $\alpha^*_{ij} = \alpha_{ij}((1 - p_{ij}) + \lambda_{ij} p_{ij})$. Notice, that there is one fading factor for each parent configuration.

Parameter estimation and adaptation can be turned on and off using the experience counts.

2.3 Inference

In the case of Bayesian networks, we define probabilistic inference as the task of computing all posterior marginals of non-evidence variables given the evidence, i.e., $P(X \mid \mathcal{E})$ for each non-evidence variable X. For influence diagrams, inference is the task of computing an optimal strategy $\hat{\Delta}$. A strategy is a set of policies, one for each decision variable in the influence diagram: The policy for decision variable D prescribes what action the decision maker should take for D given knowledge of previous observations and decisions. A strategy is said to be optimal if it maximizes the expected utility.

Before inference can take place, the Hugin Decision Engine first compiles the Bayesian network or influence diagram. The result of compilation is a structure, known as a junction tree. The nodes of the junction tree are sets of variables of the original network. These sets are sometimes called cliques because they correspond to cliques (maximal complete sets) in some triangulated graph obtained from the original network. A probability potential (and a utility potential in the case of influence diagrams) are associated with each clique and each separator between two adjacent cliques.

Inference is performed using a message passing algorithm on the junction tree. This process is known as propagation. Inference involves the following steps:

- Each item of evidence must be incorporated into the junction tree potentials. For each item of evidence, some potential, containing the variable in question, is modified to reflect the evidence.

- Some clique of the junction tree is selected. This clique is referred to as the root of the propagation.

- Then messages are passed towards the selected root. The messages are passed through the separators of the junction tree (i.e., along the links of the tree). These messages cause the potentials of the receiving cliques and separators to be updated. This phase is known as CollectEvidence.

- Now messages are passed in the other direction (i.e., from the root towards the leaves of the junction tree). This phase is known as DistributeEvidence.

- At this point, the junction tree is said to be in equilibrium: The probability $P(X \mid \mathcal{E})$ can be computed from any clique or separator containing X—the result will be independent of the chosen clique or separator.

See Lauritzen and Spiegelhalter (1988), Andersen et al. (1989), Jensen et al. (1990), Dawid (1992), Jensen et al. (1994), and Lauritzen and Jensen (2001) for details.

The junction tree structure is also suitable for many other inference tasks:

- The probability of the evidence $\mathsf{P}(\mathcal{E})$ is computed as part of a propagation. It is the normalization constant of the probability potential of the root after CollectEvidence has completed.
- The most probable explanation is the configuration of all variables that has the highest probability given the evidence. This configuration can be identified by a so-called max-propagation, which differs from a normal propagation only by using maximization instead of summation (Dawid, 1992).
- The joint probability distribution over a set of variables can be computed by combining potentials in a subtree of the junction tree containing all the variables in question.
- A variant propagation algorithm, known as fast retraction, computes, for each evidence variable X, the conditional probability given the evidence on all variables except X (Cowell and Dawid, 1992).
- Sampling of a configuration of all variables according to the conditional distribution given the evidence can be done by a simple distribution algorithm on the junction tree (Dawid, 1992).
- For networks containing conditional Gaussian (CG) variables, the normal output of a propagation consists of the mean and variance of each CG variable given the evidence. However, it is also possible to get an exact representation of the distribution of a CG variable. This distribution is a mixture of Gaussians, and an exact representation of the distribution contains a list of the parameters describing each component of the mixture (Lauritzen and Jensen, 2001).
- A conflict measure can be defined as:

$$\mathrm{conf}(\mathcal{E}) = \frac{\prod_{i=1}^{n} \mathsf{P}(\mathcal{E}_i)}{\mathsf{P}(\mathcal{E})}.$$

  If $\mathrm{conf}(\mathcal{E}) > 1$, then this is an indication of a possible conflict as we assume individual pieces of evidence to be positively correlated. The conflict is easily computed as part of a propagation (Jensen et al., 1991).

In addition to the above mentioned operations, the Hugin Decision Engine provides the user with the following special features:

- An approximation method that changes very small probabilities to zeros is supported. Zeros in the probability potentials allow the potentials to be represented using sparse tables. This can result in huge speed-ups (Jensen and Andersen, 1990).
- A suite of heuristic as well as optimal triangulation methods are available to the user.
- Parallel processing for inference on multiprocessor systems is supported. The parallel processing is implemented through parallel table operations.
- Sampling using either the Bayesian network or the junction tree structure is supported. Sampling of discrete, CG, and mixture of CG and discrete variables is supported. A configuration of the variables can be sampled according to the distribution determined by the evidence. Sampling in an uncompiled Bayesian network is possible when the set of nodes with evidence form an ancestral set of instantiated nodes. Similarly, sampling in influence diagrams is supported when all decisions are instantiated.
- Both hard and soft evidence is supported. Hard evidence is the usual "a variable is known to be in a certain state," whereas soft evidence is a likelihood function over the variable in question.

- Computations are usually performed using a single precision floating point representation, but double precision is also an option. Double precision is useful in the case of mixed Bayesian networks and when performing sequential learning.

- Special "save-to-memory" functionality is provided to support efficient initialization (i.e., establishing the initial distribution, with no evidence incorporated, on the junction tree).

3 Hugin Graphical User Interface

The Hugin Graphical User Interface provides the user with an easy-to-use interface to the advanced Hugin Decision Engine.

Using the Hugin Graphical User Interface, it is possible to create new models either by hand, from data, or using both manual and automatic model creation in combination.

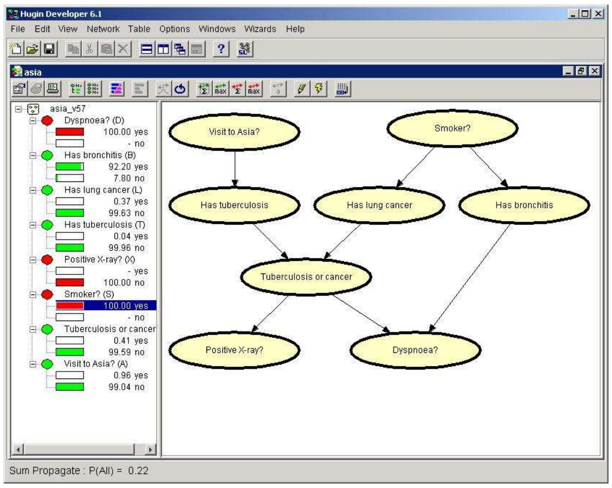

Figure 7: The Chest Clinic example.

Figure 7 shows the Hugin Graphical User Interface with the Chest Clinic example loaded with observations on Dyspnoea (yes), Positive X-ray (no), and Smoker (yes).

3.1 Table Generator

The Hugin Graphical User Interface supports many aspects of table manipulation. This includes extended support for using expressions. In the next version (v6.2) of the tool, the specification of tables through expressions (as opposed to specifying each cell manually) will be improved significantly.

The ability to specify expressions instead of entire conditional probability distributions is very powerful and can save a lot of time.

Example 3.1 [Noisy-OR]
If, e.g., the interaction model between a variable and its three parents X, Y, and Z can be modeled as a Noisy-OR interaction model, this can be written as:

NoisyOR (X, 0.3, Y, 0.5, Z, 0.2, true, 0.93).

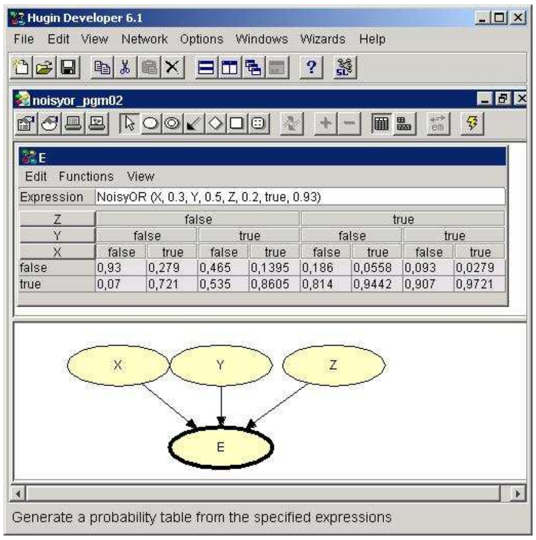

Figure 8: A Noisy-OR model.

This expression specifies a Noisy-OR model where the inhibitor probabilities are 0.3, 0.5, and 0.2 for X, Y, and Z, respectively, with a leak probability of 0.07, see Figure 8. □

The Table Generator contains a lot of other features, e.g., specification of probabilities using bars (and dragging) instead of numbers, exporting of tables to various formats (e.g., LaTeX), auto normalization, etc.

3.2 Object-Oriented Modeling

As something unique, the Hugin Tools support the concept of object-oriented model construction. This makes it simple to reuse models and encapsulate data. In particular, the reusing of models, can save a lot of work in areas where the same class of objects are instantiated numerous times.

Example 3.2 [Stud Farm, Continued]
The probability of a given horse carrying a hereditary disease can often be determined from the probability of each parent carrying the disease. This can be represented as a Bayesian network. However, if the model for each horse gets just a little complicated, it can be tiresome to create all the models, and the resulting model may be rather cluttered. Both issues are solved, when using object-oriented Bayesian networks. Figure 9 shows a model containing the relationship between the 10 different horses. Each horse is simply represented as an instance of a "horse" class, so the resulting model is quite simple.

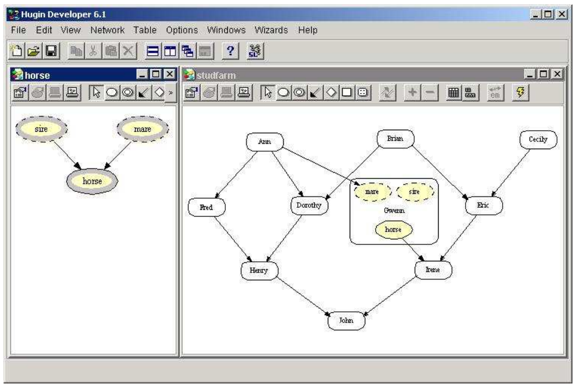

Figure 9: The Stud Farm network.

Another feature of object-oriented Bayesian networks is modularity. If e.g., one of the horses is a special breed it can easily be represented using a different model, as long as the interfaces of the the object classes are identical (the interface of the "horse" class can be seen on the expanded node in Figure 9). □

3.3 The Learning Wizard

The simplest way to create a new model is by using the Learning Wizard on a database of cases extracted from the domain one wishes to model. Using the Learning Wizard, a new model is created in just 8 steps, including accessing the data in files or databases, data preprocessing, specifying prior knowledge on both structure and distributions, and performing the learning steps.

The newest feature in the Learning Wizard is the NPC algorithm, which improves on the existing PC algorithm. As mentioned above, the NPC algorithm produces a PDAG with ambiguous regions. The user chooses which dependences in the ambiguous regions to include/exclude (or the user can choose to let the algorithm decide automatically).

Example 3.3 [Learning Wizard]
Figure 10 shows the PDAG with ambiguous regions generated by the NPC algorithm applied on a sample of $10,000$ cases from the Chest Clinic network and with a significance level of $\alpha = 0.05$.

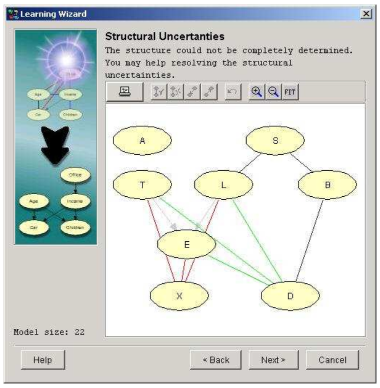

Figure 10: A PDAG with ambiguous regions for the Chest Clinic example.

In Figure 10 the ambiguous regions are shown as groups of undirected edges of the same color. Notice that some certain edges are directed and some are undirected. □

4 Concluding Remarks

Numerical stability and performance are major issues in the implementation of the Hugin Decision Engine. A lot of effort has been and will be put into achieving as high numerical stability and performance as possible.

At the moment, the set of planned future features of the Hugin Decision Engine and Hugin Graphical User Interface include, but is not limited to:

- Lazy evaluation (Madsen and Jensen, 1999).

- LImited Memory Influence Diagrams (LIMIDs) (Lauritzen and Nilsson, 2001).
- An implementation of d-separation based on the separation algorithm described in (Lauritzen et al., 1990).

A free demo-version of the Hugin Tools can be downloaded from our web-site (Hugin Decision Engine v5.4 and Hugin Graphical User Interface v6.1): http://www.hugin.com.

References

S. K. Andersen, K. G. Olesen, F. V. Jensen, and F. Jensen. 1989. HUGIN — a Shell for Building Bayesian Belief Universes for Expert Systems. In *Proc. of the 11th IJCAI*.

R. G. Cowell and A. P. Dawid. 1992. Fast retraction of evidence in a probabilistic expert system. *Statistics and Computing*, 2:37–40.

R. G. Cowell, A. P. Dawid, S. L. Lauritzen, and D. J. Spiegelhalter. 1999. *Probabilistic Networks and Expert Systems.* Springer-Verlag.

A. P. Dawid. 1992. Applications of a general propagation algorithm for probabilistic expert systems. *Statistics and Computing*, 2:25–36.

R. A. Howard and J. E. Matheson. 1981. Influence Diagrams. In *The Principles and Applications of Decision Analysis*, volume 2, chapter 37, pages 721–762.

F. Jensen and S. K. Andersen. 1990. Approximations in Bayesian Belief Universes for Knowledge Based Systems. In *Proceedings of the Sixth Workshop on Uncertainty in Artificial Intelligence*, pages 162–129.

F. V. Jensen, S. L. Lauritzen, and K. G. Olesen. 1990. Bayesian updating in causal probabilistic networks by local computations. *Computational Statistics Quarterly*, 4(2):269–282.

F. V. Jensen, B. Chamberlain, T. Nordahl, and F. Jensen. 1991. Analysis in HUGIN of data conflict. In *Uncertainty in Artificial Intelligence*, pages 519–528.

F. Jensen, F. V. Jensen, and S. Dittmer. 1994. From Influence Diagrams to Junction Trees. In *Proc. of the 10th UAI*, pages 367–373.

S. L. Lauritzen and F. Jensen. 2001. Stable Local Computation with Mixed Gaussian Distributions. *Statistics and Computing*, 11(2):191–203.

S. L. Lauritzen and D. Nilsson. 2001. Representing and solving decision problems with limited information. *Management Science.*, 47:1238–1251.

S. L. Lauritzen and D. J. Spiegelhalter. 1988. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society, B.*, 50(2):157–224.

S. L. Lauritzen, A. P. Dawid, B. N. Larsen, and H. G. Leimer. 1990. Independence properties of directed Markov fields. *Networks*, 20(5):491–505.

S. L. Lauritzen. 1992. Propagation of probabilities, means and variances in mixed graphical association models. *Journal of the American Statistical Association*, 87(420):1098–1108.

S. L. Lauritzen. 1995. The EM algorithm for graphical association models with missing data. *Computational Statistics & Analysis*, 19:191–201.

A. L. Madsen and F. V. Jensen. 1999. Lazy propagation: A junction tree inference algorithm based on lazy evaluation. *Artificial Intelligence*, 113(1-2):203–245.

C. Meek. 1995. Causal inference and causal explanation with background knowledge. In *Proc. of the 11th UAI*, pages 403–410.

K. G. Olesen, S. L. Lauritzen, and F. V. Jensen. 1992. aHUGIN: A System Creating Adaptive Causal Probabilistic Networks. In *Proc. of the 8th UAI*, pages 223–229.

J. Pearl. 1988. *Probabilistic Reasoning in Intelligence Systems.* Series in Representation and Reasoning. Morgan Kaufmann Publishers.

H. Raiffa. 1968. *Decision Analysis.* Addison-Wesley, Reading, MA.

D. Spiegelhalter and S. L. Lauritzen. 1990. Sequential updating of conditional probabilities on directed graphical structures. *Networks*, 20:579–605.

P Spirtes, C Glymour, and R Scheines. 2000. *Causation, Prediction, and Search.* Adaptive Computation and Machine Learning. MIT Press, Cambridge, Massachusetts, second edition.

H. Steck. 2001. *Constraint-Based Structural Learning in Bayesian Networks using Finite Data Sets.* Ph.D. thesis, "Institut für der Informatik der Technischen Universität München".