Proceedings of the 2007 IEEE Symposium on
Foundations of Computational Intelligence (FOCI 2007)

Learning Bayesian Network Structures with Discrete Particle Swarm Optimization Algorithm

Heng Xing-Chen, Qin Zheng, Tian Lei, Shao Li-Ping

School of Electronics and Information Engineering, Xi'an Jiaotong University, Xi'an 710049, China.

***Abstract*-A novel structure learning algorithm of Bayesian networks (BNs) using particle swarm optimization (PSO) is proposed. For searching in structure spaces efficiently, a discrete PSO algorithm is designed in term of the characteristics of BNs. Firstly, fitness function is given to evaluate the structure of BN. Then, encoding and operations for PSO are designed to provide guarantee of convergence. Finally, experimental results show that this PSO based learning algorithm outperforms genetic algorithm based learning algorithm in convergence speed and quality of obtained structures.**

I. INTRODUCTION

Bayesian network is known as probabilistic network, Bayesian belief network or causal network. It combines graph theory with probability to express complex uncertainty among random variables. Bayesian Network has been developed well as a kind of uncertain reference method. It has been implemented in applications in areas such as medical diagnostics, classification systems and software agents for personal assistants, multisensor fusion, and legal analysis of trials [1]. Until recently, the standard approach to constructing belief networks was a labour-intensive process of eliciting knowledge from experts. Methods for capturing available data to construct a Bayesian network or to refine an expert-provided network promise to greatly improve both the efficiency of knowledge engineering and the accuracy of the models. For this reason, learning Bayesian networks from data has become an increasingly active area of research.

Learning a Bayesian network can be decomposed into the problem of learning the graph structure and learning the parameters. An obvious choice to combat the problem of "getting stuck" on local maxima is to use a stochastic search method [2,3].

Several heuristic searching techniques, such as greedy hill-climbing, simulated annealing and GA have been used. Among those heuristics, GA and evolution computation [10,11] have been intensively researched and proved being effective in learning Bayesian networks. However, there exist two drawbacks of GA in learning Bayesian networks which are its expensive computational cost and premature convergence. When the number of variables in Bayesian networks is large, those drawbacks would degrade performance of the learning algorithm and make it tend to return a network structure which is local optimal.

This paper explores the use of particle swarm optimization (PSO) algorithms for learning Bayesian networks. Our choice was partly motivated by the work of Clerc et al. [12] and a discrete PSO algorithm is designed in term of the characteristics of BNs. BN structures are amenable for the discrete PSO algorithm since the substructures of the network behave as building blocks so we can evolve higher fit structures by exchanging substructures of parents with higher fitness.

II. PROBLEM FORMULATION

Bayesian networks and associated schemes constitute a probabilistic framework for reasoning under uncertainty that in recent years has gained popularity in the community of artificial intelligence [13,14]

From an informal perspective, Bayesian networks are directed acyclic graphs (DAGs), where the nodes are random variables, and the arcs specify the independence assumptions that must be held between the random variables.

To specify the probability distribution of a BN, one must give prior probabilities for all root nodes (nodes with no predecessors) and conditional probabilities for all other nodes, given all possible combinations of their direct predecessors. These numbers in conjunction with the DAG, specify the BN completely. The joint probability of any particular instantiation of all n variables in a BN can be calculated as follows:

$$P(x_1, x_2, ..., x_n) = \prod_{i=1}^{n} P(x_i \mid \pi_i) \qquad (1)$$

where, x_i represents the instantiation of the variable X_i and π_i represents the instantiation of the parents of X_i.

The most common approach to building Bayesian networks is to elicit knowledge from an expert. This works well for smaller networks, but when the number of variables becomes large, elicitation can become a tedious and time-consuming affair. There may also be situations where the expert is either unwilling or unavailable.

Whether or not experts are available, if there are data it makes sense to use it in building a model. The problem of learning a Bayesian network from data can be broken into two components: learning the structure, B_S and learning the parameters, B_P. If the

structure is known1 then the problem reduces to learning the parameters. If the structure is unknown, the learner must first find the structure before learning the parameters (actually in many cases they are induced simultaneously). Generally, it is difficult for experts to give the structure of BN directly, and learning it from data is a feasible modeling method. In addition, the structure learning algorithm can itself be decomposed into searching for structures by using search algorithm and evaluating structures by using scoring metric (fitness function), which aims at finding the best structure with highest accuracy of generating training set. So the problem of the structure learning can be formally defined as follows:

Input: A training set D of instances of **X**, which contains N observation sequences. The length of the k^{th} sequence is l_k and each case $x_k[0], x_k[1],\ldots, x_k[l_k]$ is given.

Output: A BN that best matches D. The notion "best matches" is defined using a scoring function.

III. STRUCTURE LEARNING ALGORITHM USING DISCRETE PSO

Particle swarm optimization (PSO) is an evolutionary computation technique developed by Dr. Eberhart and Dr. Kennedy in 1995, which is inspired by social behavior of bird flocking and fish schooling [2,3]. It has been found to be extremely effective in solving the continuous optimization problem, but now it has been expanded to discrete domain.

In 1997, Eberhart and Kennedy proposed a discrete binary version of PSO [4], KE-PSO, in which the discrete binary variables are operated and trajectories are changes in the probability that a coordinate will take on a zero or one. Then a multi-phase discrete particle swarm optimization is presented, in which the discrete binary version is improved and different groups of particles have trajectories that proceed along trajectories with differing goals in different phases of the algorithm [5]. In 2004, a quantum particle swarm optimization algorithm is proposed, QPSO, in which the discrete binary version is improved again by introducing the quantum theory [6]. In 2005, a Bayesian optimization model-oriented approach to particle motion algorithm is proposed in the literature [7], where the use of an explicit information model as the basis for particle motion provides tools for designing successful algorithms. Though some problems still exist, it can be easily modified for any discrete/combinatorial problem for which we have no good specialized algorithm. Therefore, as a combinatorial optimization problem, it is possible to learn the structure of BN by using PSO algorithm.

Our algorithm which is named after "BNDP" can be expressed simply by the following equation, BNDP = (F, X, V, S_{xx}, P_{vv}, M_v, P_{xv},, λ, G_{init}, υ), where, F is a fitness function, X a space of positions of particles, V velocity set of particles, S_{xx} a substraction operation (position, position), P_{vv} a move operation (position plus velocity), M_v a multiplication operation (coefficient times velocity), P_{xv} a addition operation (velocity plus velocity), λ the swarm size, G_{init} an initial swarm and υ stopping condition.

A. *Fitness Function F*

The most common fitness function *F* to evaluating structures is by the posterior probability of the structure given the observations. That is, a structure is good to the extent it is probable given the available information. The posterior probability of a structure can be obtained by applying Bayesian rule:

$$P\ (B_S \mid D) = \frac{P\ (D \mid B_S)\ P\ (B_S)}{P\ (D)} \qquad (2)$$

where $P(B_S|D)$ is the posterior distribution of the structure given the data, $P(D|B_S)$ is the likelihood function, $P(B_S)$ is the prior probability of the structure, and P(D) is the normalizing constant. Since P (D) is not dependent on the structure, it can be ignored when trying to find the best scoring function. In addition, without prior knowledge of structures, we can assume they have equal probability. However, if we do have information on structures we can always use the prior information.

The problem is now reduced to finding the structure with the maximum likelihood $P(D|B_S)$. In other words, given a structure, these structures are evaluated according to how probable it is that the data were generated from the structure.

Cooper and Herskovitz showed that when a Dirichlet prior is used for the parameters in the network, the likelihood $P(D|B_S)$ can be obtained in closed form:

$$\mathrm{F} = \mathrm{P}(\mathrm{D} \mid \mathrm{B_S}) = \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{\Gamma(N'_{ij})}{\Gamma(N'_{ij} + N_{ij})} \prod_{k=1}^{r_i} \frac{\Gamma(N'_{ijk} + N_{ijk})}{\Gamma(N'_{ijk})} \qquad (3)$$

where n is the number of variables in the database, r_i is the number of possible states for variable X_i , q_i is the number of possible states for pa (X_i), N_{ijk} are the sufficient statistics from the database (counts of occurrences of configurations of variables and their parents), N'_{ijk} are the hyper parameters (prior counts of occurrences of variables and their parents) specified for the parameter prior (assuming an uninformative prior as in the prior for the structure we set the hyper parameters to 1), $N_{ij} = \sum_{k=1}^{r_i} N_{ijk}$, and $N'_{ij} = \sum_{k=1}^{r_i} N'_{ijk}$. For computational convenience, the number of parents allowed for a particular variable is limited. This scoring metric (3) is commonly referred to as the Bayesian Dirichlet metric. In practice, the logarithm of

(3) is usually used to score networks.

When data are complete, the Bayesian Dirichlet metric, a fitness function for BN, exists in closed form. So we may utilize the score decomposition properties, which facilitate the computation of the scoring metric (3) in several ways. Note that the likelihood is expressed as a sum of terms, where each term depends only on the conditional probability of a variable given a particular assignment to its parents. Thus, if we want to find the maximum likelihood parameters, we can maximize within each family independently.

With the Bayesian Dirichlet metric (see Eq. (3)), we can now search over possible structures for the one that scores best networks from complete datasets that mean that all of the cases in the data contain values for all of the variables.

B. *Encoding PSO Elements for BN*

The BN structure can be represented as an adjacency list, see Fig. 1, where each row represents a variable X_i and the members of each row, with the exception of the first member, are the parents of X_i, pa (X_i). The first member of each row, i.e. the first column of the adjacency list, is the variable X_i.

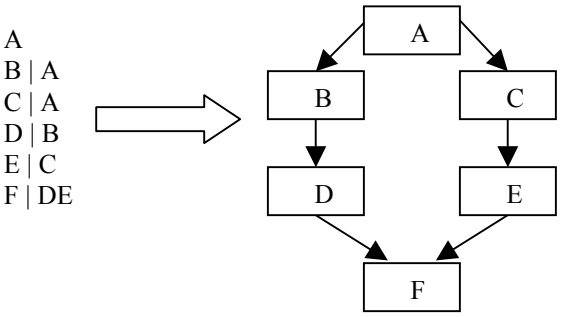

Fig.1. Encoding the structure of a BN

• *Position of Particles and State space*

As BNDP algorithm is designed to find the best structure of BN by using PSO, the structure of BN should be encoded into a position of particle.

Although we show the variable X_i in the Fig. 1 for clarity, the internal representation encodes its parents only, with the variable being encoded by sequence. The adjacency list can be thought of as a "position" where each pa (X_i) represents a "local position". For example, the "local position" of the variable F can be encoded as [D, E]. Because the logarithm of the scoring metric is the summation of scores for each variable, each local position can be scored separately and added to generate the fitness score for the entire structure.

So the search space is enormous. A local position can range from no parents to n-1 parents, where n is the number of variables in the dataset. Thus a local position can take on $\sum_{i=1}^{k}\binom{n-1}{i}$ possible values where k is the maximum set of parents a variable can have and n is the number of variables in the dataset. So, the search space can be defined as follow: $\sum_{j=1}^{n}\sum_{i=1}^{k}\binom{n-1}{i}$.

• *Velocity*

Definition 1 ***switch operator***
If a specified BN has n variables, the position of a particle can be expressed as an adjacency list P = ((x_i)), i=1,…,n. We define a switch operator *SO* which, when applied to a position during one time step, gives another position. So, here, *SO* has three types: $+x_i$, $-x_i$ and ϕ. The $+x_i$ denotes adding a variable x_i into original position, $-x_i$ reducing a variable x_i and ϕ null.

Definition 2 ***switch unit***
A sequence composed of one or several switch operators is a switch unit. We denote it by SU.

$$SU = (SO_1, SO_2, ., SO_k) \quad (4)$$

where $SO_1, SO_2, ., SO_k$ are k switch operators and the different orderings of them have different signification. The length of the SU is defined by ||SU|| =k. Each SU is applied to the change of a local position.

Definition 3 ***switch list (velocity)***
A sequence composed of one or several switch units is a switch list. We denote it by SL. Here, actually, the number of switch units of each switch list is equal to the number of variables *n* of BN. The length of the SL is defined by $||SL|| = \sum_{i=1}^{n} || SU_i ||$. A velocity *V* is then defined by

$$V = SL = (SU_1, SU_2, ., SU_n) \quad (5)$$

Where, $SU_1, SU_2, ., SU_n$ are n switch units and each SL or *V* is applied to the change of a global position.

Definition 4 *equivalent set of switch list*
If different switch lists are equivalent (same result when applied to any position), the set of them is called equivalent set of switch list.

C. *Designing Operations for PSO*

• *Opposite of a velocity*
It means to do the same switch as in original *SL*, but with reverse operator. For example, - ((-A), (+B-C)) = ((A), (-B+C)). It is easy to verify that we have - (- *SL*) =*SL* (and $SL \oplus -SL \cong \phi$, see

below Addition "velocity plus velocity").

• *Addition (P_{xv}) "position plus velocity"*
Let P be a position and V a velocity. The position $P' = P \dot{+} V$ is found by applying the first switch of V to P, then the second one to the result etc.
Example

$$\begin{cases} P = (\phi, A, A, B, C, DE) \\ V = ((+B), (-A), (-A+B), (-B+C), (\phi), (-D-E+A)) \end{cases} \quad (6)$$

Applying V *to* P, we obtain successively

$$P' = (B, \phi, B, C, C, A) \quad (7)$$

• *Substraction (S_{xx}) "position minus position"*
Let P_1 and P_2 be two positions. The difference $P_2 - P_1$ is defined as the velocity V, found by a given algorithm, so that applying V to P_1 gives P_2. The condition "found by a given algorithm" is necessary, for, as we have seen, two velocities can be equivalent, even when they have the same size. In particular, the algorithm is chosen so that we have $P_1 = P_2 \Rightarrow V = P_2 - P_1 = \phi$

• *Addition (P_{vv}) "velocity plus velocity"*
Let V_1 and V_2 be two velocities. In order to compute $V_1 \oplus V_2$ we consider the switch list which contains the first switch unit of V_1, followed by the first switch unit of $V2$, then the second switch unit of V_1, followed by the second switch unit of $V2$ etc. For example, ((-A), (-B+C)) $\oplus$ ((-B+A), (-B+D)) = ((-A-B+A), (-B+C-B+D)). In general, we "contract" it to obtain a smaller equivalent velocity. For example, ((-A-B+A), (-B+C-B+D)) = ((-B), (+C+D)). In particular, this operation is defined so that $V \oplus -V = \phi$. So, we can have the following definition:

Definition 5 *basic switch list*
The switch list of equivalent set of switch list, which contains the least switch operators, is defined as basic switch list. Each velocity is a basic switch list.

• *Multiplication (M_v) "coefficient times velocity"*
Let α be a real coefficient and V be a velocity. There are different cases, depending on the value of α.
Case $\alpha = 0$
We have $\alpha V = \phi$
Case $\alpha \in [0,1]$
We just "shrink" V. Let $\|\alpha V\|$ be the greatest integer smaller than or equal to $\alpha \|V\|$. So we define $\alpha V = ((SO_1, .., SO_k)_1, \ldots, (SO_1, .., SO_k)_n), k \uparrow_1^{(\|\alpha V\|/n)}$
Case $\alpha > 1$
It means we have $\alpha = d + \alpha'$, d is an integer ($d \neq 0$), $\alpha' \in [0,1]$.
So we define $\alpha V = \sum_{i=1}^{d} (\oplus V) \oplus \alpha' V$.
Case $\alpha < 0$
As $\alpha V = (-\alpha) * (-V)$, we only need to consider one of the previous cases.

D. *Control Parameters*

The initial swarm G_{init}, as well as the velocities, can be generated either randomly or by a Sobol sequence generator [9, 15], which ensures that the D-dimensional vectors will be uniformly distributed within the search space.

The swarm size λ should be not kept too big because of the computation time required scoring the fitness function; On the other hand, λ should be not kept too small for improving the diversity of particles of swarm to avoid premature convergence. Hence, we choose λ within [30,100].

The stopping criterion υ for the algorithm is set in term of that when either g_1 generations have been run or when in g_2 successive generations, the value of the fitness function of the best structure corresponds with the average value of the fitness function.

E. *BNDP Algorithm*

We can now rewrite the formula from the basic PSO algorithm:

$$V_{id}^{k+1} = w * V_{id}^{k} \oplus c_1 * rand() * (P_{id} - X_{id}^{k}) \oplus c_2 * Rand() * (P_{gd} - X_{id}^{k}) \quad (8)$$

$$X_{id}^{k+1} = X_{id}^{k} + V_{id}^{k+1} \quad (9)$$

Where, i = 1,2,…,N; N is the swarm's size; d represents the d-dimensional search space; w is the inertia weight factor; c_1 and c_2 are two positive constants, called the cognitive and social parameter respectively; $rand()$ and $Rand()$ are two random numbers uniformly distributed within the range [0,1]; V_{id}^{k} is the velocity of particle i at iteration k; X_{id}^{k} is the current position of particle i at iteration k; P_{id} is the best previous position of particle i at iteration k; P_{gd} is the best neighbour's best previous position at iteration k.

The BNDP algorithm can be described as follows:

Step1: Initialize the particle swarm (each particle is given a stochastic initial solution/position and switch list/velocity).
Step2: If stopping criterion is satisfied, turn to *Step 5*.

Step 3: Calculate the next position X'_{id} (the new solution) according to the current position X_{id} of the particle i.

1) Calculate the difference α by $\alpha = P_{id} - X_{id}$ where α is a basic switch list and is applied to X_{id} to obtain P_{id}.
2) Calculate the difference β by $\beta = P_{gd} - X_{id}$ where β is also a basic switch list.
3) Calculate the velocity V'_{id} in term of the equation (8) and transform V'_{id} into a basic switch list.
4) Calculate the new solution X'_{id} in term of the equation (9).
5) If a better solution is found, update P_{id}.

Step 4: If a better solution is found for the whole swarm, update P_{gd} and turn to *Step 2*.

Step 5: Show the optimal solution.

Ⅳ. EXPERIMENT

For evaluating the behavior of our algorithm BNDP, we perform the different experimental steps as follows:

Step 1: Begin with a BN (structure + conditional probabilities) and simulate it, generating randomly 1000 samples for the training set D and another 1000 for the test set.

Step 2: Using the approach based on BNDP algorithm try to obtain best BN structure B^*_S from D, which maximize the probability $P(D|B^*_S)$.

Step3: Evaluate the performance of BNDP algorithm by evaluating the accuracy of B^*_S predicting objective probability distribution.

For this experiment we use a Bayesian network known as ASIA. The ASIA network was initially presented by Lauritzen and Spiegelhalter [8]. It is a small (nine variables) fictitious model of medical knowledge concerning the relationships between visits to Asia, tuberculosis, smoking, lung cancer, lung cancer or tuberculosis, Positive X-ray, Dyspnoea and bronchitis. The training set with 1000 samples is generated using probabilistic logic sampling [15] from the original network.

We compare BNDP algorithm with structure learning algorithm based on GA presented in [10]. The algorithm proposed in [10] is named "BNGA" in this paper. The best structure of BN over 10 times running BNDP and the BNGA algorithm on the training set is selected as the finally obtained result model. We evaluate the performance of the algorithms by using the accuracy of predicting the probability distribution of the objectives through the result model. This concrete process is that we generate 1000 samples as test set from the original network, then calculate the log loss $(\frac{1}{N_{num}}\sum_{i=1}^{N_{num}}\log p(X^i[0],...,X^i[N_i]))$ for the test set using the result model, which can be seen in figure 2.

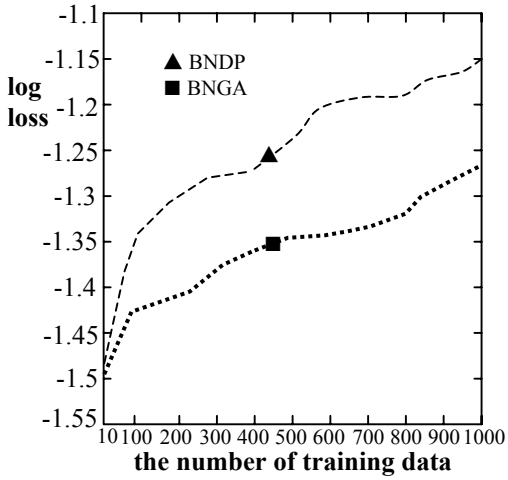

Fig.2. Comparison of Performance of algorithms

From figure 2 we could see that, for all training sets, the predictive accuracy of BNDP is much higher than that of BNGA. And compared with BNGA, the more the number of samples is introduced, the higher the predictive accuracy of our algorithm becomes. In addition, the average number of iterations of BNDP (476) is much smaller than that of BNGA (644) for 1000 samples during the search process. The reason for our algorithm's superiority on "efficiency" over BNGA is that, PSO can converge much more rapidly than GA. Therefore PSO is very promising for learning Bayesian networks.

Ⅴ. CONCLUSION

In this paper we introduce PSO to the problem of learning Bayesian networks from data. This problem is characterized by a large, multi-dimensional, multi-modal search space and is extremely difficult for deterministic algorithms. We propose an efficient structure learning algorithm using discrete PSO, which is called BNDP algorithm.

Using simulations of the ASIA network, we carry out a performance analysis on the BNDP algorithm. The obtained experimental results also show that, compared to structure learning algorithm using GA, the BNDP algorithm could reduce the time for learning Bayesian network greatly. In addition, the

quality of the finally obtained network structure could be also improved. Therefore, discrete PSO is a very promising for learning Bayesian networks when the number of variables is very large.

The future step forwards is to extend BNDP algorithm for learning the structure of dynamic Bayesian networks from incomplete data.

ACKNOWLEDGEMENT

The research reported in this paper is funded by the National "973" Key Basic Research Development Planning Project (2004CB719401).

REFERENCES

[1] Heckerman, D., D. Geiger, et al. (1995). "Learning Bayesian Networks: The Combination of Knowledge and Statistical Data." Machine Learning 20:197-243.

[2] Kennedy, J. Minds and cultures: particle swarm implications. *Socially Intelligent Agents: Papers from the 1997 AAAI Fall Symposium* pp. 67-72. AAAI Press, Menlo Park, CA, 1997. usa, pp.289-294.

[3] Eberhart, R. C. and J. Kennedy. 1995. A new optimizer using particle swarm theory. *Proceeding of the sixth International symposium on micro machine and human science* pp. 39-43. IEEE service center, Piscataway, NJ, Nagoya, Japan.

[4] J. Kennedy and R.C. Eberhart, "A discrete binary version of the particle swarm algorithm," Proc. Conf. On Systems, Man, and Cybernetics, Piscataway, NJ, 1997.

[5] Al-Kazemi B, Mohan C K. "Multi-phase generalization of the particle swarm optimization algorithm." In: Proceedings of the 2002 Congress on Evolutionary Computation, CEC' 02,Vol.1,2002: 489~494

[6] Shuyuan Yang, Min Wang, "A Quantum Particle Swarm Optimization," In: Proceedings of the 2004 Congress on Evolutionary Computation, CEC'04,Vol.1,2004:320~324.

[7] Christopher K. Monson, Kevin D. Seppi, "Bayesian Optimization Models for Particles Swarms", Proc. Seventh Genetic and Evolutionary Computation Conference, Washington, DC, USA, June 25–29, 2005.

[8] Lauritzen, S.L. and D.J. Spiegelhalter. 1988. "Local Computations with Probabilities on Graphical Structures and Their Application on Expert Systems," *J. Royal Statistical Soc. B*, vol. 50, no. 2, pp. 157-224.

[9] W.H. Press, W.T. Vetterling, S.A. Teukolsky and B.P. Flannery, Numerical Recipes in Fortran 77,Cambridge University Press: Cambridge, 1992.

[10] Larraaga, P., Poza, M., Yurramendi, Y., Murga, R., Kuijpers, C.: Structural learning of Bayesian network by genetic algorithms: performance analysis of control parameters. IEEE Trans. Pattern Anal.Machine Intell. 18 (1996) 912–926.

[11] LI, X.L., Yuan, S.M., He, X.D.: Learning Bayesian Networks Structures Based on Extending Evolutionary Programming. Proceeding of the Third International Conference on Machine Learning and Cybernetics. Shanghai (2004) 1594–1598.

[12] Clerc, M., Discrete Particle Swarm Optimization, illustrated by the Traveling Salesman Problem, New Optimization Techniques in Engineering, Springer, 2004, 219-239

[13] Pearl, J. 1998. Probabilistic Reasoning in Intelligence Systems: Network of Plausible Inference. San Mateo, Calif.: Morgan Kaufmann.

[14] Neapolitan, R.E. 1990. Probabilistic Reasoning in Expert Systems. Theory and Algorithms. John Wiley & Sons.

[15] Henrion, M. 1988. "Propagating Uncertainly in Bayesian Networks by Probabilities Logic Sampling," *Uncertainty in Artificial Intelligence*, vol. 2, pp. 149-163.