A Genetic Algorithm for Tuning Variable Orderings in Bayesian Network Structure Learning

Haipeng Guo Benjamin B. Perry Julie A. Stilson William H. Hsu

Laboratory for Knowledge Discovery in Databases, Kansas State University
234 Nichols Hall, Manhattan, KS 66506-2302
{hpguo | bhsu | bbp9857 | jas3466}@cis.ksu.edu http://www.kddresearch.org

1. Introduction

In the last two decades or so, Bayesian networks (BNs) [Pe88] have become a prevalent method for uncertain knowledge representation and reasoning. BNs are directed acyclic graphs (DAGs) where nodes represent random variables, and edges represent conditional dependence between random variables. Each node has a conditional probabilistic table (CPT) that contains probabilities of that node being a specific value given the values of its parents. The problem of learning a BN from data is important but hard. Finding the optimal structure of a BN from data has been shown to be *NP*-hard [HGC95], even without considering unobserved or irrelevant variables. In recent years, many Bayesian network learning algorithms have been developed. Generally these algorithms fall into two groups, score-based search and dependency analysis (conditional independence tests and constraint solving). Many previous approaches require that a node ordering is available before learning. Unfortunately, this is usually not the case in many real-world applications. To make greedy search usable when node orderings are unknown, we have developed a permutation genetic algorithm (GA) wrapper to tune the variable ordering given as input to *K2* [CH92], a score-based BN learning algorithm. In our continuing project, we have used a probabilistic inference criterion as the GA's fitness function and we are also trying some other criterion to evaluate the learning result such as the learning fixed-point property.

2. K2 and GA Wrapper

K2 uses a Bayesian score and a greedy search method to construct a Bayesian network from a database of records [CH92]. It can find structures quickly given a reasonable variable, or node, ordering. Given an ordering, only "upstream" nodes of a node are considered as its candidate parents. *K2* is very sensitive to the ordering because of its limitation of greediness. The search scheme we have implemented is to use a permutation genetic algorithm as a wrapper to explore the ordering space by randomly generating, selecting, and recombining node orderings. This GA applies *K2* using each node ordering, and estimates which among the BN structures output by *K2* is most probable. To develop a GA for permutation problems, we need to select the proper crossover and mutation operations. We use order crossover (*OX*) for this task. *OX* exchanges subsequences of two permutations, displacing duplicated indices with holes. It then shifts the holes to one side, possibly displacing some indices, and replaces the original subsequence in these holes. Mutation is implemented by swapping uniformly selected indices.

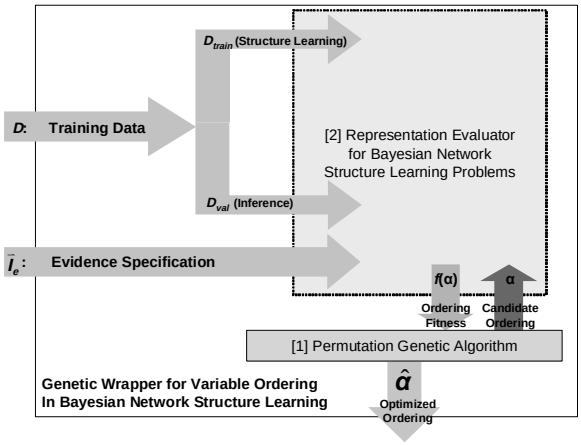

Figure 1. System Design Overview.

After the GA produces a random ordering, we validate it via inference using the BN produced by *K2*. The problem of BN inference is to compute *P(Q|E)*, the posterior probabilities of query nodes given states of evidence nodes. BN inference is also *NP*-hard [Co90]. For small and sparse networks, we can use exact inference algorithm to compute *P(Q|E)*. The Lauritzen-Spiegelhalter *(LS)* algorithm, or clique-tree propagation, is one efficient exact BN inference algorithm [LS88]. For large and

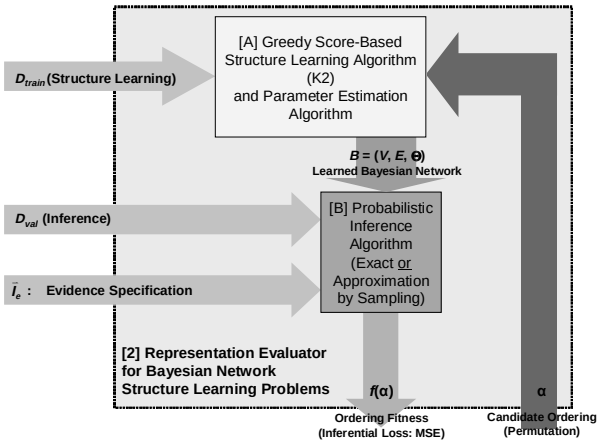

Figure 2. Probabilistic reasoning environment, Module [2] from Figure 1.

complex BNs, we apply approximate inference. Stochastic simulation algorithms are the most often used approximate inference methods, which include logic sampling, likelihood weighting, backward sampling, self-importance sampling, and adaptive importance sampling *(AIS)*. We divide the input data into training data D_{train} and validation data D_{val}. D_{train} is used to learn the BN and D_{val} is used to validate the learned BN. We first compute $P(Q|E)$ from D_{val}, and $P'(Q|E)$ from the BN produced by K2, using either LS or AIS. We then compute a loss function (RMSE) between $P(Q|E)$ and $P'(Q|E)$, and use it as the GA's fitness function. Figure 1 shows the system overview.

3. Result & Discussion

System components implemented so far include *K2* for learning BNs, the LS algorithm and several stochastic sampling algorithms for inference, and the GA wrapper for tuning the node ordering. We implemented an elitist permutation GA using OX. Our initial experiments were conducted on an eight nodes BN whose permutation space contains 8! = 40320 different orderings. The result shows the GA (10 populations, 100 generations) improves the ordering to within 0.01 of the optimal RMSE (about 0.95 calculated by LS algorithm on the standard BN). Our work is closely related to [LPYM+96] in which they also applied a GA to Bayesian network learning. But they only focused on structure learning. They used the same Bayesian score as *K2* to evaluate a learned network structure and they used GA to directly operate on the network structure itself. Our scheme uses a different fitness function and the GA operates on a smaller space, i.e., the variables ordering space. Although our initial result is promising, we feel that it could be further improved by trying better fitness functions for the GA wrapper. The problem we need to overcome is the same as discussed in [NK00]: "In case where the amount of data is small relative to the size of the model, there are likely to be many models that explain the data reasonably well". This makes us less confident about that the learned model is a true representation to the underlying process. To improve confidence we are considering sample complexity issues in the learning process. We are also trying to use the fixed-point property of the learning process as a fitness function to evaluate the learned model. Our conjecture is that a good initial model shall converge in probability to a fixed point under a learning and stochastic data generation process while a bad model shall not. Our preliminary result shows that this may be a promising direction in the future.

Acknowledgements

Support for this research was provided in part by the Army Research Lab under grant ARL-PET-IMT-KSU-07 and by the Office of Naval Research under grant N00014-01-1-0519.

References

[CD00] J. Cheng and M. J. Druzdzel. AIS-BN: An adaptive importance sampling algorithm for evidential reasoning in large Bayesian networks. *Journal of Artificial Intelligence Research (JAIR)*, 13:155-188, 2000.

[CH92] G. F. Cooper and E. Herskovits. A Bayesian Method for the Induction of Probabilistic Networks from Data. *Machine Learning*, **9**(4):309-347, 1992.

[Co90] G. F. Cooper. The computational complexity of probabilistic infernece using bayesian belief networks. *Artificial Intelligence*, 42(2-3):393-405. Elsevier, 1990.

[HGC95] D. Heckerman, D. Geiger, and D. Chickering, Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20(3):197-243, Kluwer, 1995.

[LPYM+96] P. Larranaga, M. Poza, Y. Yurramendi, R. H. Murga and C.M.H. Kuijpers. Structure Learning of Bayesian Networks by Genetic Algorithms: A Performance Analysis of Control Parameters. *IEEE Journal on Pattern Analysis and Machine Intelligence* 18(9): 912-926, 1996.

[LS88] S. L. Lauritzen and D. J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society, Series B* 50, 1988.

[NK00] N. Friedman and D. Koller. Being Bayesian About Network Structure: A Bayesian Approach to Structure Discovery in Bayesian Networks. In UAI00.

[Pe88] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann, San Mateo, CA, 1988.