Probabilistic Reasoning Through Genetic Algorithms and Reinforcement Learning

Xiaomin Zhong and **Eugene Santos Jr.**
Department of Computer Science and Engineering
University of Connecticut
Storrs,CT 06269-3155
zhongx@cse.uconn.edu and eugene@cse.uconn.edu

Abstract

In this paper, we develop an efficient approach for inferencing over Bayesian networks by using a reinforcement learning controller to direct a genetic algorithm. The random variables of a Bayesian network can be grouped into several sets reflecting the strong probabilistic correlations between random variables in the group. We build a reinforcement learning controller to identify these groups and recommend the use of "group" crossover and "group" mutation for the genetic algorithm based on these groupings. The system then evaluates the performance of the genetic algorithm and continues with reinforcement learning to further tune the controller to search for a better grouping.

Introduction

Bayesian Networks (Pearl 1988a) are one of the most popular models for uncertainty. Knowledge is organized in a hierarchical fashion providing easy visualization of the reasoning domain. Such networks consist of directed acyclic graphs of nodes, each representing a random variable(rv) with a finite domain. The directed arcs between the nodes represent probabilistic conditional dependencies. The joint probability over the random variables can be computed via the chain rule and the given conditional independence assumption. Bayesian networks have been applied to various domains such as story comprehension, planning, circuit fault detection and medical diagnoses.

There are two types of computations performed with Bayesian networks: belief updating and belief revision(Pearl 1988b). Belief updating concerns the computation of probabilities over random variables, while belief revision concerns finding the maximally probable global assignment. However, both tasks are known to be NP-hard (Shimony 1994). In this paper, we demonstrate how a kind of genetic algorithms directed by a reinforcement learning controller can be effectively used in belief revision.

A Genetic Algorithm(GA) is an evolutionary computation technique inspired from the principles of natural selection to search a solution space. Most researches modified their implementation of GA either by using non-standard chromosome representation or by designing problem specific genetic operations(Michalewizs 1996) to accommodate the problem to be solved, thus building efficient evolution programs. In this paper, we employ "group" crossover and "group" mutation based on grouping random variables of a Bayesian Network(BN).

Experimental results show that different groupings effect the performance of the GA. We use a Reinforcement Learning(RL) controller to adaptively determine the elements in each group. This method investigates the impact of using domain knowledge during the recombination and mutation phases of the GA.

Bayesian Networks and Belief Revision

Belief revision is the process of determining the most probable instantiation of the random variables (rvs)in a network given some evidence. More formally, if W is the set of all rvs in the given Bayesian network and e is the evidence (that is, e represents a set of instantiations made on subset of W), any complete instantiations to all the rvs in W that is consistent with e is called an explanation on interpretation of e. The problem is to find an explanation w^* such that: $p(w^*) = \max_{w \in W} p(w|e)$ Intuitively, we can think of the non-evidence rvs in W as possible hypotheses for e.

With a small network, a valid solution method is to simply tabulate all the possible values of the rvs and then calculate the probabilities. Once the network gets larger and more complex, this method is obviously unacceptable and more efficient method must be employed.

Several people have used GAs to perform belief revision over Bayesian network, but only superficially considered the topological structure of BN (Rojas-Guzman & Kramer 1993; Santos, Shimony, & Williams 1997; Santos & Shimony 1998; Welch 1996). From experimental results, we know that even the topological structure will effect the performance of GA (Williams, Santos, & Shimony 1997; Jitnah & A.E.Nicholson 1996).

So we use reinforcement learning to continuously learn and identify the problem-specific attributes of the BN, and employ this in the GA through "group" crossover and "group" mutations for the search process.

Genetic Algorithm Approach

Genetic algorithms are search and optimization algorithms based on the principles of natural evolution. To apply GA, one generally expresses the problem in such a way that potential solution can be coded in a gene-like bit sequence and a population of those sequence is prepared. An optimal solution is searched for by evolutionary operations including selection, crossover and mutation. Most researchers modified their implementations of GA either by using nonstandard chromosome representation or by designing problem specific genetic operations to accommodate the problem to be solved in order to build efficient evolution programs.

When using GA to implement the process of belief revision on BN, we develop "group" operations for this particular problem. The GA used in our method is based on the GENESIS(Grefenstette 1990) framework. In this section, we will describes the representation and operations employed in our GA.

Representation For the belief revision problem domain, we represent a solution or individual which is a complete assignment to the underlying BN as an array of integers (the gene). Each position in the array (chromosome) corresponds to one random variable in belief network. Each element of the array can take a number of values from a finite discrete integers and corresponds to the number of the random variables in BN. The genetic operations manipulate each individual by changing the value of each element in the array. The fitness evaluation is simply our solution quality calculation. In this case, it is the joint probability of the solution.

Selection The selection operation is the standard "roulette wheel" selection approach based on the ranking of the individuals within the population instead of the absolute performance. Ranking helps prevent premature convergence by preventing "super" individuals from taking over the population within a few generations.

Crossover The crossover operation performs a "group" crossover. We partition the random variables of BN into several groups based initially on the topological connectivity (see RL later). Each group is a subset of the random variables that forms a "connected" subgraph of the BN. We hope a rv will be in the same group with its parents and children. In this paper, a reinforcement learning controller is used to identify grouping.

The mechanism of the "group" crossover is explained by the following examples. Assume there is a BN with 12 random variables. The 12 random variables have been appropriately divided into 3 groups:group1={1,2,8,12}, group2={3,6,7,9}, and group3={4,5,10,11}. Each group represent a subgraph. Assume, there are two selected parents p_1 = (123311223213) and p_2 = (221313113211). For subgraph G_1 (group1), subgraph G_2 (group2) and subgraph G_3, the subsolution can be calculated:

$$p(G_1) = \prod_i p(x_{1-i}|\pi(x_{1-i})) \quad (1)$$

$$p(G_2) = \prod_j p(x_{1-j}|\pi(x_{2-j})) \quad (2)$$

$$p(G_3) = \prod_k p(x_{1-k}|\pi(x_{3-k})) \quad (3)$$

where $\pi(\ )$ is the set of instantiations to the parents of the random variables.

For p_1, using the above formulas, we get subsolution $p(G_1^1)$, $p(G_2^1)$, $p(G_3^1)$. For p_2, we get $p(G_1^2)$, $p(G_2^2)$, $p(G_3^2)$. We then compare $p(G_1^1)$ with $p(G_1^2)$, $p(G_2^1)$ with $p(G_2^2)$, and $p(G_3^1)$ with $p(G_3^2)$. Let's assume $p(G_1^1) > p(G_1^2)$, $p(G_2^1) > p(G_2^2)$ and $p(G_3^1) < p(G_3^2)$. We can get two children p_1' and p_2' by combining the good subsolutions together:

$$p(p_1') = p(G_1^1)p(G_2^1)p(G_3^2) \quad (4)$$

and

$$p(p_2') = p(G_1^2)p(G_2^1)p(G_3^2) \quad (5)$$

where p_1' = (123311223213) and p_2' = (223311213211).

Mutation There are two kinds of mutation operation employed in our method. One is standard mutation and the other is called "group" mutation. Standard mutation randomly selects a chromosome to modify and then randomly choose a new value for that chromosome. "Group" mutation randomly select a chromosome to update the group it belongs to. These mutation operations helps the GA to maintain diversity in the population to avoid premature convergence.

Reinforcement Learning System for Grouping Identification

As described above, we use GAs to implement the task of belief revision. In order to improve performance, we introduced "group" crossover and "group" mutation. RL is used to identify the elements of each group. In this section, we introduce the structure and learning algorithms of RL employed in our system.

Reinforcement Learning

Reinforcement learning is different from supervised learning. At each time step in supervised learning, a teacher provides the desired control objective to the learning system. In reinforcement learning, the teacher's response is not as direct and informative as in supervised learning and it serves more to evaluate the state of the system. In general, reinforcement learning is more widely applicable than supervised learning, since many control problems require selecting control actions whose consequences emerge over uncertain periods for which input-output training data are not readily available.

In the reinforcement learning scheme, inputs describing the environment is given to the learning system.

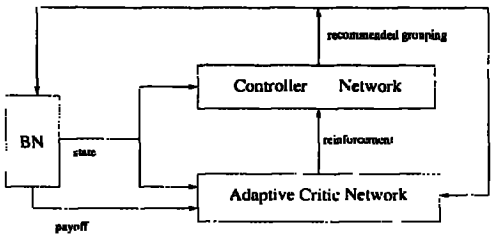

FIG. 0.1. Reinforcement Learning

The system then makes a decision according to these inputs, thereby causing the environment to deliver a reinforcement to the system. The reinforcement measures the decision made by the system. In some system, the system receives reinforcement after each decision, that is, a complete evaluation of an action becomes available before the system needs to perform the next action. In some systems, however, reinforcement is often temporally delayed. The reinforcement learning algorithms employed in our system is similar to the method proposed by (Santharam & Sastry 1997).

Architecture Figure 0.1 shows the architecture of our proposed learning system:

The system has two major parts: the controller and the adaptive critic. The controller makes a stochastic choice of rv groupings in each state based on its internal parameter. This parameter is updated at each instant using a scalar reinforcement signal supplied by the adaptive critic network. The adaptive critic network maintains estimates of state-action values (Santharam & Sastry 1997) and supplies a reinforcement for the controller. Here the state of a rv describes which group the rv belongs to. For example, if a rv's state is j, the rv belongs to group j. The action k acting on a rv means changing the rv's state to group k. For a rv i and action k, the state-action value q_{ik} is defined as the expected infinite horizon total discounted payoff if action k is performed and an optimal policy is followed thereafter.

Controller network The structure of the controller is as follows: It has one unit corresponding to each rv of Bayesian network. Each unit is a learning automata. Figure 0.2 show the structure of the kth unit:

The kth unit of the controller has a weight vector $w_k = (w_{k1}, \ldots, w_{kn})$. $g_{ki} \in G$, $G = \{1, 2, \ldots, n\}$ is the group state set. The unit generates an group output $y_k \in G$ stochastically according to the law: $y_k = i$ *with probability* p_{ki} and

$$p_{ki} = \frac{f(w_{ki}g_{ki})}{\sum_{m=1}^{n} f(w_{km}g_{km})} \tag{6}$$

where $f : R \to R$ is the activation function given by $f(x) = \frac{1}{1+e^{-x}}$. From the above formula it follows that each value of parameter w_k determines a specific stationary random grouping.

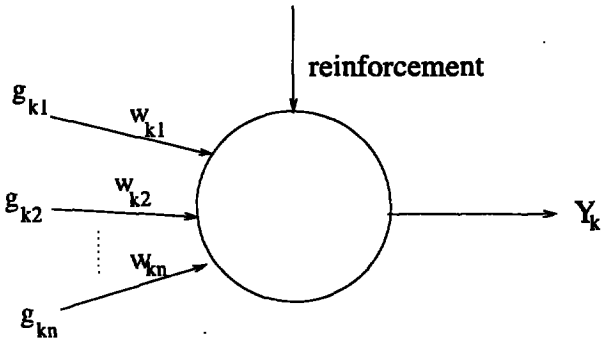

FIG. 0.2. Structure of kth Controller Unit

Adaptive Critic Network The function of the adaptive critic network is to update the estimates of the state-action values after observing the state and payoff. It has one unit corresponding to each rv of BN. There are two tasks are implemented in each unit. First, according to the immediate payoff value, estimate each current state-action value q_{ki}. Second, choose the maximum state-action value among the all state-action values.

Learning Algorithm The learning algorithm is as follow:

1. For $k = 1$ to n (n is the number of rvs in BN) the kth controller unit generates an action y_k using equation 6.
2. For k=1 to n
   (a) the reinforcement r' to the kth controller unit from the kth adaptive critic will be:

$$r' = \begin{cases} 1 & q_{ki} = \max_{m \in G} q_{km} \\ 0 & \text{otherwise} \end{cases}$$

   (b) update the kth controller weights:

$$\Delta w_{ki} = \beta(r' - f(w_{ki})) - \delta w_{ki} \tag{7}$$

   where β is the learning rate, δ is the weight decay rate. The weight decay is to prevent converging prematurely.

   (c) update the state-action value:

$$\Delta q_{ki} = \beta(r(n) - q_{ki} + \alpha \max_{m \in G} q_{km}) \tag{8}$$

   where β is the learning rate, α is the discount factor, and r(n) is the payoff at instant n.

The above steps are repeated at each time instant.

Combining GA with RL

As stated above, the GA is used to implement belief revision, and the RL is used to identify groupings for the GA. The system proceeded in two steps: initial grouping selection and run-time control. The goal of initial grouping selection is to learn the structure of the BN through RL and identify crossover groupings. In run-time control, the GA interacts with the RL controller in order to tune the GA performance and find a better grouping that results in improved GA performance.

Initial grouping selection

For initial grouping, we desire that a rv will be in the same group with its parents and children. For instant n, we use a ratio $E_1(n)$ to evaluate the grouping result produced by the RL controller:

$$E_1 = \frac{number\ of\ good\ rv}{number\ of\ all\ rv\ in\ BN} \tag{9}$$

A good rv means that the rv is in the same group with its parents and children. After the RL controller generates a grouping, E_1 is computed, and payoff:

$$r(n) = \frac{E_1(n)}{E_1(n-1)} \tag{10}$$

is sent back to the RL controller to change the parameters of RL controller.

Run-time Control

In this phase, the GA interacts with the RL controller, and tunes each other. The RL controller sends its grouping results based on initial grouping selection to the GA. The GA implements its operation (selection, crossover and mutation) to find a solution. A payoff is produced to tune the RL controller. The payoff r(n) must also consider the performance of the GA. Based on the grouping, for each group we compute subsolution of the best individual of the population, and use the best subsolution E_2 to determine the performance of the GA. At instant n, we use the following to compute the payoff:

$$r(n) = w_1 \frac{E_1(n)}{E_1(n-1)} + w_2 \frac{E_2(n)}{E_2(n-1)} \tag{11}$$

From our experiments, we have found that values of $w_1 = 0.1$ and $w_2 = 0.9$ to be ideal.

Experiments

We tested our approach on several Bayesian networks. Four of these, which we called trace1, trace2, trace3 and trace4, were generated by the Andes (Gertner, Conati, & VanLehn 1998) system. The others were randomly generated with different network topologies and distributions. We ran both the standard GA and our GA through 600 iterations. In cases where possible, we determined the optimal solution through brute force computations.

Table 0.1 shows performance comparison of *trace* networks, and Figure 0.3 shows the results running our GA and the standard GA on these *trace* networks.

Table 0.2 shows the results on BNs generated randomly with different probability distributions and connective density, where *exp1* has exponential probability distribution and a dense connection, *exp2* has exponential probability distribution and a sparse connection, *flat1* has flat probability distribution and a dense connection, and *flat2* has flat probability distribution and a sparse connection. The last network *spike-flat* has a mix of exponential and flat distribution nodes. Figure 0.4 depicts the performance of our GA and the standard GA on just the exponential BNs.

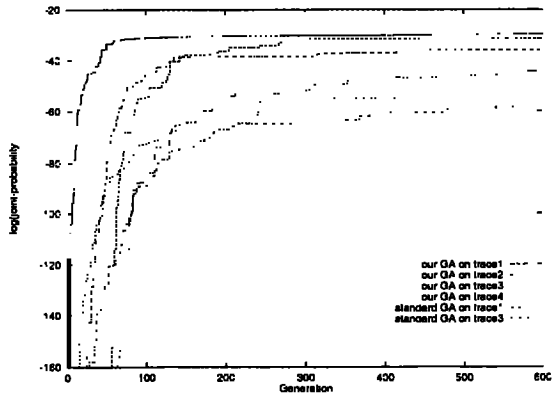

FIG. 0.3. Performance on *trace* network

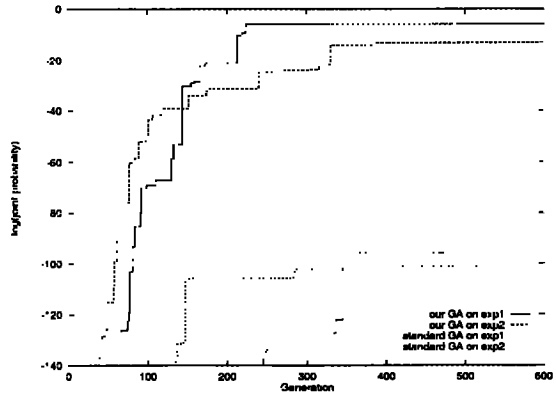

FIG. 0.4. Performance on BN with *exponential* distribution

Finally, Table 0.3 shows the results on BNs generated randomly with different numbers of nodes but with similar exponential probability distributions and connective density.

From our experiments, we can see that our new GA with group crossover and mutation performs significantly over the standard GA approach. Intuitively, our approach to capture groups of highly correlated random variables can be seen in the networks with exponential distributions (near 0-1 probabilities). However, as the last experiment indicated, there are still limitations with the approach as the search space becomes explo-

BN name	Node	Optimal solution	Standard GA	Our GA
BN1	100	2.650e-01	0	2.65e-01
BN2	200	3.601e-02	0	1.305-05
BN3	300	8.13e-04	0	1.767e-04
BN4	400	n/a	0	0

TABLE 0.3. Performance Comparison of BN with different *node number*

BN name	Node	Optimal solution	Standard GA	Time (CPU Secs)	Our GA	Time (CPU Secs)
trace1	103	4.032e-11	6.548e-16	58.5	2.199e-13	117.91
trace2	158	9.186e-12	0.0	n/a	1.207e-14	192.27
trace3	186	7.026e-18	7.089e-26	141.62	1.467e-24	238.38
trace4	196	n/a	0.0	n/a	1.912e-20	257.90

TABLE 0.1. Performance Comparison on *trace* BNs.

BN name	Node	Optimal solution	Standard GA	Time (CPU Secs)	Our GA	Time (CPU Secs)
exp1	100	8.463e-02	1.810e-40	59.85	5.396e-06	118.45
exp2	100	1.960e-01	1.565e-35	55.33	4.938e-04	110.71
flat1	100	5.948e-23	3.362e-25	62.00	5.445e-24	112.38
flat2	100	8.037e-26	8.783e-27	56.66	1.654e-27	108.20
spike-flat	60	2.172e-12	9.166e-26	29.57	7.538e-18	67.2

TABLE 0.2. Performance Comparison of BN with different *distribution*

sive. In general, we have traded additional computational overhead in our method for orders of magnitude improvement in our solution.

Conclusion

For large multiply connected networks, exact inferencing may not be feasible, making approximate algorithms an attractive alternative. We have proposed a new GA with "group" crossover and "group" mutation for inferencing over BN. A reinforcement learning controller is used to identify the grouping. During inferencing, the controller interacts with the GA. With this interaction, the GA and the controller tune each other to find good solutions. This method investigates the impact of using domain knowledge during the recombination and mutation phases of GA. Preliminary results have show that our proposed method constitute a promising approach to perform inference in multiply connected complex system. This method can yields sub-optimal solution in tractable times. In general, we have traded additional computational overhead in our method for orders of magnitude improvement in our solution. While the approach has limitations such as actually determining the optimal solution, the sub-optimal solution can be used in cooperation with other algorithms to improve their performance by taking a portfolio-based approach to problem solving (Williams, Santos, & Shimony 1997).

Acknowledgements This work was supported in part by AFOSR Project #940006 and #9600989.

References

Gertner, A. S.; Conati, C.; and VanLehn, K. 1998. Procedural help in Andes: Generating hints using a Bayesian network student model. In *Proceedings of the Fourteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI-98)*. San Francisco, CA: Morgan Kaufmann Publishers.

Grefenstette, J. J. 1990. *A user's guide to GENESIS Version 5.0.* Navy Center for Applied Research in AI.

Jitnah, N., and A.E.Nicholson. 1996. Belief network inference algorithms: a study of performance based on domain characterisation. Technical Report TR-96-249, Monash University, Clayton, VIC, 3168 Australia.

Michalewizs, Z. 1996. *Genetic Algorithms + Data Structure = Evolution Programs.* New York: Springer-Verlag.

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

Pearl, J. 1988b. *Probability Reasoning in Intelligent System: Networks of Plausible Inference.* San Mateo, CA: Morgan Kaufmann Publishers.

Rojas-Guzman, C., and Kramer, M. A. 1993. GALGO: A Genetic ALGOrithm decision support tool for complex uncertain systems modeled with bayesian belief networks. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 368–375.

Santharam, G., and Sastry, P. 1997. A reinforcement learning neural networks for adaptive control of markhov chains. *IEEE Transactions on System, Man and Cybernetics-PartA: Systems and Humans* 72:558 - 600.

Santos, Jr., E., and Shimony, S. E. 1998. Deterministic approximation of marginal probabilities in bayes nets. *IEEE Transactions on Systems, Man, and Cybernetics* 28(4):377–393.

Santos, Jr., E.; Shimony, S. E.; and Williams, E. 1997. Hybrid algorithms for approximate belief updating in bayes nets. *International Journal of Approximate Reasoning* 17(2-3):191–216.

Shimony, S. E. 1994. Finding MAPs for belief networks is NP-hard. *Artificial Intelligence* 68:399–410.

Welch, R. L. 1996. Real time estimation of bayesian networks. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence.*

Williams, E.; Santos, Jr., E.; and Shimony, S. E. 1997. Experiments with distributed anytime inferencing: Working with cooperative algorithms. In *Proceedings of the AAAI Workshop on Building Resource-Bounded Reasoning Systems*, 86–91.