Structure Learning of Bayesian Networks Using a Semantic Genetic Algorithm-Based Approach

Sachin Shetty
Department of Electrical and Computer Engineering
Old Dominion University
Norfolk, VA 23529, USA
sshetty@odu.edu

Min Song
Department of Electrical and Computer Engineering
Old Dominion University
Norfolk, VA 23529, USA
msong@odu.edu

Abstract **A Bayesian network model is a popular technique for data mining due to its intuitive interpretation. This paper presents a semantic genetic algorithm (SGA) to learn a complete qualitative structure of a Bayesian network from a database. SGA builds on recent advances in the field and focuses on the generation of initial population, crossover, and mutation operators. Particularly, we introduce two semantic crossover and mutation operators that aid in faster convergence of the SGA. The crossover and mutation operators in SGA incorporate the semantic of the Bayesian network structures to learn the structure with very minimal errors. SGA has been proved to perform better than existing classical genetic algorithms for learning Bayesian networks. We present empirical results to prove the fast convergence of SGA and the predictive power of the obtained Bayesian network structures.**

Index Terms— **Data mining, Bayesian networks, genetic algorithms, structure learning**

I. Introduction

One of the most important steps in data mining is to build a descriptive model of the database being mined. To do so, probability-based approaches have been considered an effective tool because of uncertain nature of descriptive models. Unfortunately high computational requirements and the lack of proper representation have hindered the building of probabilistic models. To alleviate the above twin problems, probabilistic graphical models have been proposed. In the past decade, many variants of probabilistic graphical models have been developed, with the simplest variant being Bayesian networks (BNs) [1]. Bayesian networks are employed to reason under uncertainty, with wide varying applications in the field of medicine, finance, and military planning [1] [3]. A Bayesian network is a popular descriptive modeling technique for available data by giving an easily understandable way to see relationships between attributes of a set of records. Computationally, Bayesian networks provide an efficient way to represent relationships between attributes and allow reasonably fast inference of probabilities. A lot of research has been initiated towards learning these networks from raw data as the traditional designer of a Bayesian network may not be able to see all of the relationships between the attributes.

The authors would like to thank the support from ECE Department of Old Dominion University.

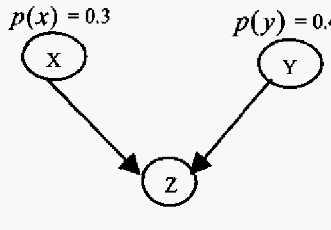
Learning and reasoning are the main tasks of analyzing Bayesian networks. We focus on the structure learning of a Bayesian network from a complete database. The database stores the statistical values of the variables as well as the conditional dependence relationship among the variables. We employ a Genetic algorithm (GA) technique to learn the structure of Bayesian networks. Genetic algorithms have been successfully applied in optimization tasks [7] [8]. The Bayesian network learning problem can be viewed as an optimization problem where a Bayesian network has to be found that best represents the probability distribution that has generated the data in a given database.

The rest of the paper is organized as follows. Section 2 introduces the background of Bayesian network and GA, and the related work in Bayesian network structure learning. In Section 3 we discuss the details of our approach for structure learning in a Bayesian network structure using a modified GA. In Section 4, we experiment two different genetic algorithms. The first one is the genetic algorithm with classical genetic operators. In the second algorithm, we extended the standard mutation and crossover operators to incorporate the semantic of the Bayesian network structures. Section 4 also presents the results for the two genetic algorithms under the two constraints. Finally, Section 5 concludes the paper and proposes future research.

II. Background and Related Work

A. *Structure learning of Bayesian networks*

Formally, a Bayesian network consists of a set of nodes which represent variables, and a set of directed edges between the nodes. The nodes and directed edges constitute a directed acyclic graph (DAG). Each node is represented by a finite set of mutually exclusive states. The directed edges between the nodes represent the dependence between the linked variables. The strengths of the relationships between the variables are expressed as conditional probability tables (CPT). A Bayesian network efficiently encodes the joint probability distribution of its variables. The encoding can also be viewed as a conditional decomposition of the actual joint probability of variables. Fig. 1 depicts a hypothetical example of a Bayesian network.



a. DAG

$p(z \mid x, y)$	0.2
$p(z \mid x, \overline{y})$	0.8
$p(z \mid \overline{x}, y)$	0.1
$p(z \mid \overline{x}, \overline{y})$	0.6

b. CPT

Figure 1. A Bayesian network with three variables

The three variables shown in Figure 1 are $X = \{x, \overline{x}\}$, $Y = \{y, \overline{y}\}$, and $Z = \{z, \overline{z}\}$, where X and Y are parents of Z. The DAG for the example is shown in Fig. 1a and the CPT for node Z is shown in Fig. 1b. The joint probabilities can be determined as $p(X,Y,Z) = p(X).p(Y).p(Z \mid X,Y)$. From the figure, it is clear that once a structure is found, it is necessary to compute the conditional probabilities for node Z with parents X and Y for all possible combinations and their prior probabilities.

In general, the DAG and CPT specify the Bayesian network and represent the distribution for the n-dimensional random variable $(X_1, \ldots, X_n)$:

$$P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i | pa(x_i))$$

where x_i represents the value of the random variable X_i and $pa(x_i)$ represents the value of the parents of X_i. Thus, the structure learning problem of a Bayesian network is equivalent to the problem of searching the optimum in the space of all DAGs. During the search process, a trade-off between the structural network complexity and the network accuracy has to be made. The trade-off is necessary as complex networks suffer from over fitting making the run time of inference very long. A popular measure to balance complexity and accuracy is based on the principle of minimal description length (MDL) from information theory [6]. In this paper the Bayesian network learning problem is solved by searching for a DAG that minimizes the MDL score.

B *Related Work*

Larranaga *et al.* proposed a genetic algorithm based upon the score-based greedy algorithm [2]. In their GA implementation, a DAG is represented by a connectivity matrix that is stored as a string (the concatenation of its rows). Recombination is implemented as one-point crossover on these strings, while mutation is implemented as random bit flipping. In a related work, Larranaga *et al.* [10] employed a wrapper approach by implementing a GA that searches for an ordering that is passed on to K2 [4], a score-based greedy learning algorithm. The results of the wrapper approach were comparable to those of their previous GA. Different crossover operators have been implemented in a GA to increase the adaptiveness of the learning problem with good results [11]. Lam *et al.* [6] proposed a hybrid evolutionary programming (HEP) algorithm that combines the use of independence tests with a quality based search. In the HEP algorithm, the search space of DAGs is constrained in the sense that each possible DAG only connects two nodes if they show a strong dependence in the available data. The HEP algorithm evolves a population of DAGs to find a solution that minimizes the MDL score. The common drawback to the algorithms proposed by Lam and Larranga is that the crossover and mutation operators they used were classical in nature and the operators do not help the evolution process to reach the best solution. Our algorithm differs from the above algorithms in the design of crossover and mutation operators.

III. MODIFIED GA APPROACH

A *Representative Function and Fitness Function*

The first task in a GA is the representation of initial population. To represent a BN as a GA individual, an edge matrix or adjacency matrix is needed. The set of network structures for a specific database characterized by n variables can be represented by an $n \times n$ connectivity matrix C. Each bit represents the edge between two nodes where

$$C_{ij} = \begin{cases} 1, & \text{if } j \text{ is a parent of } i \\ 0, & \text{otherwise} \end{cases}$$

The two-dimensional array of bits can be represented as an individual of the population by the following string $C_{11}C_{12}\ldots C_{1n}C_{21}C_{22}\ldots C_{2n}\ldots C_{n1}C_{n2}\ldots C_{nn}$, where the first n bits represent the edges to the first node of the network, and so on. It can be easily found that C_{kk} are the irrelevant bits which represent an edge from node k to itself which can be ignored by the search process.

With the representative function decided, we need to devise the generation of the initial population. There are several approaches to generate initial population. We implemented the Box-Muller random number generator to select how many parents would be chosen for each individual node. The parameters for the Box-Muller algorithm are the desired average and standard deviation. Based on these two input parameters, the algorithm generates a number that fits the distribution. For our implementation, the average corresponds to the average number of parents for each node in the resultant BN. After considerable experimentation, we found that the best average was 1.0 with a standard deviation of 0.5. Though this approach is simple, it creates numerous illegal DAG due to cyclic subnetworks. An algorithm to remove or fix these cyclic structures has to be designed. The algorithm simply removes a random edge of a cycle until cycles are not found in a DAG individual.

Now that the representative function and the population generation have been decided, we have to find a good fitness function. Most of the current state-of-the-art implementations use the fitness function proposed in the algorithm K2 [4]. The K2 algorithm assumes an ordered list of variables as its input. The K2 algorithm maximizes the following function by searching for every node from the ordered list a set of parent nodes:

$$g(x_i, p_a(x_i)) = \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{N_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} N_{ijk}!$$

where r_i represents the possible value assignments ($v_{i1}, \ldots, v_{ir_i}$) for the variable with index *i*, N_{ijk} represents the number of instances in database in which a variable X_i has value v_{ik}, and q_i represents the number of unique instantiations of $p_a(x_i)$.

B Mutation and Crossover Operators

We introduce two new operators SM (semantic mutation) and SPSC (single point semantic crossover) to the existing standard mutation and crossover operators. The SM operator is a heuristic operator that toggles the bit value of a position in the edge matrix. After the toggle process, the fitness function $g(x_i, p_a(x_i))$ is evaluated to ensure that the new individual with the new toggled value leads to a better solution. The new crossover operator is specific to our representation function. As the function is a two-dimensional edge matrix consisting of columns and rows, our new crossover operator operates on either columns or rows. Thus the crossover operator generates two offsprings by either manipulating columns or rows. The SPSC crosses two parents by a) manipulating columns or parents and maximizing the function $g(x_i, p_a(x_i))$, and b) manipulating rows or children and maximizing the function $\prod_i g(x_i, p_a(x_i))$. By combining SM and SPSC, we implement our new genetic algorithm: semantic GA (SGA). Following is the pseudocode for the semantic crossover operation. The algorithm expects an individual as input and returns the modified individual after applying semantic crossover operations.

Psuedocode for Semantic Crossover

```
Step 1. Initialization
    Read input individual and populate parent table for each node

Step 2. Generate new individual
        2.1 For each node in the individual do the following n
            times
        2.2 Execute the Box Mueller algorithm to find how
            many parents need to be altered.
        2.3 Ensure that the nodes selected as parents do not form
            cycles. If cycles are formed repeat 2.2
        2.4 Evaluate the network score of the resultant
    structure.
        2.5 If current score is higher than previous score, then
            the chosen parents are the new parents of the
            selected node.
        2.6 Repeat 2.2 through 2.5.

Step 3. Return the final modified individual.
```

IV. SIMULATIONS

In this section we present the overall simulation methodology to verify SGA. Fig. 2 shows the overall simulation setup to evaluate our genetic algorithm. The following are the main steps carried out:

Step 1. Determine a Bayesian network (structure + conditional probabilities) and simulate it using a probabilistic logic sampling technique [13] to obtain a database *D*, which reflects the conditional independence relations between the variables;
Step 2. Apply our SGA approach to obtain the Bayesian network structure B_s, which maximizes the probability $P(D | B_s)$;
Step 3. Evaluate the fitness of the solutions.

The simulations were implemented by incorporating our SGA algorithm into the Bayesian Network Tools in Java (BNJ) [9]. BNJ is an open-source suite of software tools for research and development using graphical models of probability. A learning algorithm based on SGA was implemented and incorporated into the tool kit. SGA was implemented as a separate module using the BNJ API. To depict the Bayesian network BNJ visually provides a visualization tool to create and edit the network. The Bayesian networks of different network sizes were used in the simulations. The network sizes are 8, 12, 18, 24, 30 and 36. The 8 node Bayesian network used in the simulations is from the ASIA networks [12]. The remaining networks were created by adding extra nodes to the basic ASIA network. The ASIA network introduced by Lritzen and Spiegelhater illustrate their method of propagation of evidence, considers a small of fictitious qualitative medical knowledge. Fig. 3 shows the structure of ASIA network.

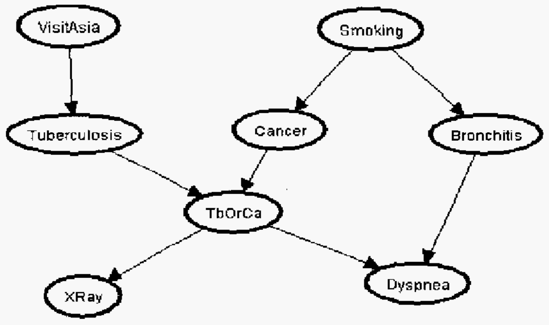


Figure 3. Structure of the ASIA network.

There are several techniques for simulating Bayesian network. For our experiments we have adopted the probabilistic logic sampling technique. In this technique, the data generator generates random samples based off the ASIA network's joint probability distribution table. The data generator sorts nodes topologically and picks a value for each root node using the probability distribution, and then generates values for each child node according to its parent's values in the joint probability table. The root mean square error (RMSE) of the data generated compared to the network it was generated from is approximately 0. This indicates that the data was generated correctly. We have populated the database with

2000, 3000, 5000 and 10,000 records. This was done to measure the effectiveness of the learning algorithm in presence of less information and more information.

The algorithm used the following input:

- Population size λ. The experiments have been carried out with $\lambda = 100$ and $\lambda = 150$.
- Crossover probability p_c, we chose $p_c = 0.9$.
- Mutation rate p_m, we considered $p_m = 0.1$.

The fitness function used by our algorithm is based on the formula proposed by Cooper and Herskovits [4]. For each of the samples (2000, 3000, 5000, 10000), we executed 10 runs with each of the above parameter combinations. We considered the following four metrics to evaluate the behavior of our algorithm.

- *Average fitness value* – This is an average of finesses function values over 10 runs.
- *Best fitness value* – This value corresponds to the best fitness value throughout the evolution of the GA.
- *Average graph errors* – This represents the average of the graph errors between the best Bayesian network structure found in each search, and the initial Bayesian network structure. Graph errors are defined to be an addition, a deletion or a reversal of an edge.
- *Average number of generations* – This represents the number of generations taken to find the best fitness function.

To evaluate SGA, we also implemented the classical genetic algorithm (CGA) with classical mutation (CM) and single point cyclic crossover (SPCC) operators. Fig. 4 plots the average fitness values for the following parameter combination. The average and best fitness values are expressed in terms of $\log P(D|B_s)$. The numbers of records are 10,000. The population size is set at 200 and the probabilities for the crossover operators SPCC and SPSC are kept at 0.9 and for mutation operators SM and CM are kept at 0.1. The figure also shows the best fitness value for the whole evolution process. One can see that SGA performs better than CGA in the initial 0-20 generations. After 20 generations, the genetic algorithm using both operators stabilizes to a common fitness value. The final fitness value is very close to the best fitness value. An important observation is that the average fitness value does not deviate a lot even after 100 generations. The best fitness value is carried over to every generation and does not get affected.

Figs. 5 and 6 plot the final learned network after the evolution process. The final learned Bayesian network was constructed from the final individual generated after 100 generations. The representation of the individual is in the form of a bit string representing an adjacency matrix. The conversion from this bit-string to the equivalent Bayesian network is trivial. Fig. 5 refers to the learned Bayesian network graph after 100 generations for 5,000 records, while Fig. 6 shows the learned Bayesian network graph for 10,000 records. It can be observed that for both the scenarios, the learned network differs from the actual generating network shown in Fig. 3 by a small number of graph errors. It is also worth noting that the numbers of graph errors reduce when the total numbers of records increase. This could mean that to reduce the total number of graph errors, a large number of records need to be provided.

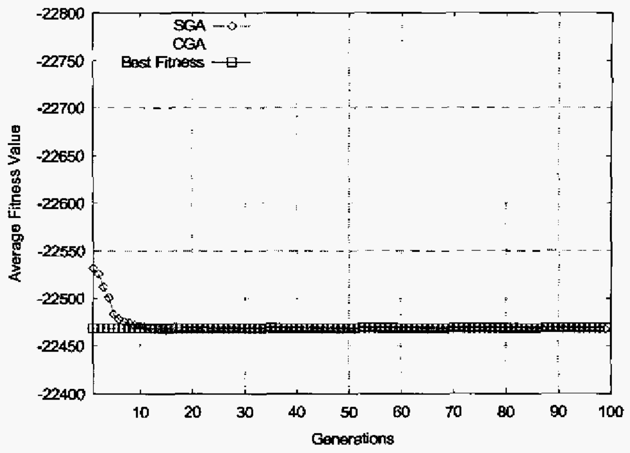


Figure 4. Plot of generations versus average fitness values (10000 Records).

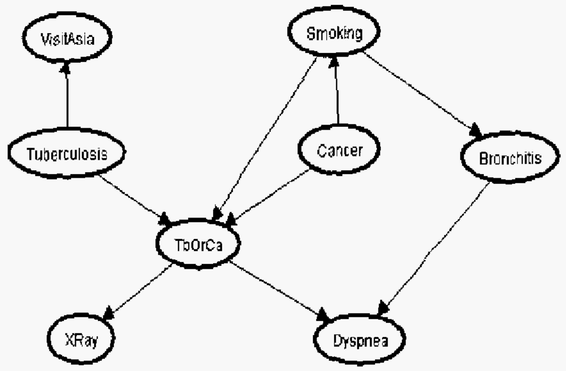


Figure 5. Learned network after 100 generations for 5,000 records - *graph errors* = 3.

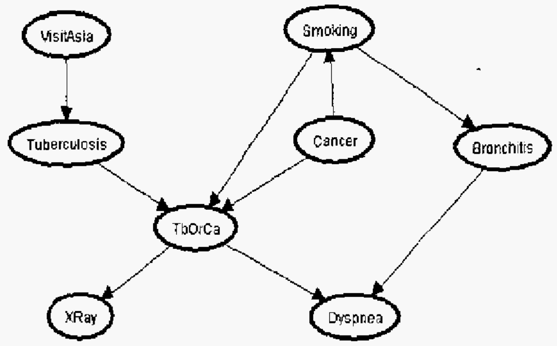


Figure 6. Learned network after 100 generations for 10,000 records - *graph errors* = 2.

Tables 1 and 2 provide the average number of generations and the average graph errors for the different number of records. It is obvious that for 2000 records the total number of generations taken to achieve the stabilized fitness value is very high. Also the average number of graph errors is too high. For 3,000, 5,000, and 10,000 records the values for the metrics are very reasonable and acceptable. To compare the performance of SGA with CGA in the presence of larger BN structures, we modified the 8-node ASIA network and generated 5 additional BN with node sizes 12, 18, 24, 30, and 36. Tables 3-7 show results for simulations carried out on these additional BNs. The tables compare the average graph errors in both approaches. The accuracy of SGA does not deteriorate under increased network sizes.

Table 1. Average number of generations

Records	SGA	CGA
2000	50	45
3000	25	30
5000	20	15
10000	20	15

Table 2. Average graph errors for 8 and 12 nodes

Records	SGA(8)	CGA (8)	SGA(12)	CGA(12)
2000	7	8	7	8
3000	3	4	3	4
5000	2	3	2	3
10000	2	3	2	3

Table 3. Average Graph errors for 18 and 24 nodes

Records	SGA(18)	CGA(18)	SGA(24)	CGA(24)
3000	3	4	3	4
5000	2	3	40	47
10000	2	3	52	61

Table 4. Average Graph errors for 30 and 36 nodes

Records	SGA(30)	CGA(30)	SGA(36)	CGA(36)
3000	53	63	60	68
5000	60	66	70	74
10000	70	72	79	81

V. Conclusions and Future Work

We have presented a new semantic genetic algorithm (SGA) for BN structure learning. This algorithm is another effective contribution to the list of structure learning algorithms. Our results show that SGA discovers BN structures with a greater accuracy than existing classical genetic algorithms. Moreover, for large network sizes, the accuracy of SGA does not degrade. This accuracy improvement does not come with an increase of search space. In all our simulations, 100 to 150 individuals are used in each of the 100 generations. Thus 10,000 to 15,000 networks are totally searched to learn the BN structure. Considering the exhaustive search space is of 2^{n^2} networks, only a small percentage of the entire search space is needed by our algorithm to learn the BN structure.

One aspect for future work is to change the current random generation of adjacency matrices for the initial population generation. The second future work is to improve scalability by parallelizing the genetic algorithm. There are two computationally expensive operations in our genetic algorithm. The first deals with initializing the first generation of individuals and evaluating fitness values for each of them. A strategy to divide the initial populations among a cluster of workstations could be devised to speed up the computational process and reduce the final running time.

References

[1] J. Pearl, "Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference," *Morgan Kaufman*, San Mateo, 1988.

[2] P. Larrañaga, M. Poza, Y. Yurramendi, R. H. Murga, and C. M. H. Kuijpers, "Structure learning of bayesian networks by genetical algorithms: a performance analysis of control parameters," *IEEE Trans. On Pattern Analysis and Machine Intelligence*, vol 18, no 9, 1996.

[3] F. V. Jensen, "Introduction to Bayesian networks," *Springer-Verlag New York, Inc.*, Secaucus, NJ, 1996.

[4] G. Cooper and E.A. Herskovits, "A Bayeisan Method for the Induction of Probabilitsic Networks from Data," *Machine Learning*, Vol. 9, No 4, pp. 309-347, 1992.

[5] D. Heckerman, D. Geiger, and D.M. Chickering, "Learning Bayesian Networks: The combination of knowledge and statistical data," *Machine Learning*, Vol. 20, pp. 197-243, 1995.

[6] W. Lam, and F. Bacchus, "Learning Bayesian Belief Networks –An Approach Based on the MDL Principle," *Computational Intelligence*, Vol. 10, No. 4, 1994.

[7] R. F. Hart, "A global convergence proof for a class of genetic algorithms," *University of Technology*, Vienna, 1990.

[8] U. K. Chakraborty, and D.G. Dastidar, "Using reliability analysis to estimate the number of generations to convergence in genetic algorithms," *Information Processing Letters*, Vol. 46, No. 4, pp. 199-209, 1993.

[9] http://bnj.sourceforge.net

[10] P. Larranaga, C. M. H. Kuijpers, R. H. Murga, and Y. Yurramendi, "Learning Bayesian network structures by searching for best ordering with genetic algorithm," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 26, No. 4, pp. 487-493, 1996.

[11] C. Cotta, and J. Muruzabal, "Towards more efficient evolutionary induction of Bayesian networks," *Proceedings of the Parallel Problem Solving from Nature VII*, pp. 730-739. Springer-Verilag, 2002.

[12] S. L. Lauritzen, and D. J. Spiegelhalter, "Local computations with probabilities on graphical structures and their application on expert systems," *J. Royal Stat.. Soc. B*, Vol. 50, No.2, pp. 157-224, 1988.

[13] M. Henrion, "Propagating uncertainty in Bayesian networks by probabilistic logic sampling," *Uncertainty in Artificial Intelligence* 2, pp. 149-163, 1988.