A Hybrid Approach to Discover Bayesian Networks From Databases Using Evolutionary Programming

Man Leung Wong
Department of Computing and Decision Sciences
Lingnan University
Tuen Mun, Hong Kong
mlwong@ln.edu.hk

Shing Yan Lee
Department of Computer Science and Engineering, CUHK,
Shatin, Hong Kong
sylee@cse.cuhk.edu.hk

Kwong Sak Leung
Department of Computer Science and Engineering, CUHK,
Shatin, Hong Kong
ksleung@cse.cuhk.edu.hk

Abstract

This paper describes a novel data mining approach that employs evolutionary programming to discover knowledge represented in Bayesian networks. There are two different approaches to the network learning problem. The first one uses dependency analysis, while the second one searches good network structures according to a metric. Unfortunately, both approaches have their own drawbacks. Thus, we propose a novel hybrid algorithm of the two approaches, which consists of two phases, namely, the Conditional Independence (CI) test and the search phases. A new operator is introduced to further enhance the search efficiency. We conduct a number of experiments and compare the hybrid algorithm with our previous algorithm, MDLEP [18], which uses EP for network learning. The empirical results illustrate that the new approach has better performance. We apply the approach to a data sets of direct marketing and compare the performance of the evolved Bayesian networks obtained by the new algorithm with the models generated by other methods. In the comparison, the induced Bayesian networks produced by the new algorithm outperform the other models.

1 Introduction

Conventional business research is a process in which data are analyzed manually to explore the relationships among various factors defined by the researcher. Even with powerful computers and versatile statistical software, many hidden and potentially useful relationships may not be recognized by the analyst. Nowadays, such problems are more acute as many businesses are capable of generating and collecting a huge amount of data in a relatively short period. The explosive growth of data requires a more efficient way to extract useful knowledge. Thus, business research is a major area for applying data mining that aims at discovering novel, interesting, and useful knowledge from databases [4]. Through data mining, researchers can discover complex relationships among various factors and extract meaningful knowledge to improve the efficiency and quality of managerial decision making. In this paper, we propose a novel data mining approach that employs Evolutionary Programming (EP) to discover knowledge represented in Bayesian networks and apply the approach to handle the business problem of finding response models from direct marketing data.

A Bayesian network is a graphical representation that depicts conditional independence among random variables in the domain and encodes the joint probability distribution [13]. With a network at hand, probabilistic inference can be performed to predict the outcome of some variables based on the observations of others. Therefore, Bayesian networks are often used in diagnostic systems [8].

Typically, a Bayesian network is constructed by eliciting knowledge from domain experts. To reduce imprecision due to subjective judgments, researchers start to be interested in constructing a Bayesian network from collected data or past observations in the domain. In the literature, there are two main approaches to this network learning problem [3]. The first one is the dependency analysis approach [3, 17]. Since a Bayesian network describes conditional independence, we could make use of dependency test results to construct a Bayesian network that conforms to our findings. The second one, called the score-and-search approach [7, 6, 10], uses a metric to evaluate a candidate network structure. With the metric, a search algorithm is employed to find a network structure which has the best score. Thus, the learning problem becomes a search problem. Unfortunately, the two approaches both have their own drawbacks. For the former approach, an exponential number of dependency tests have to be performed. Moreover,

some test results may be inaccurate [17]. For the latter approach, since the search space is huge, some Bayesian network learning algorithms [7] adopt greedy search heuristics which may easily make the algorithms get stuck in a local optimum [6].

In this work, a hybrid approach is developed for the network learning problem. Simply put, dependency analysis results are used to reduce the search space of the score-and-search process. With such reduction, the search process would take less time for finding the optimal solution. Together with the introduction of a new operator and some modifications of our previous work, MDLEP [18], we call our new approach HEP (hybrid EP). HEP is found to have the best results in a real-life application of direct marketing amongst similar state-of-the-art approaches.

This paper is organized as follows. In section 2, we present the backgrounds of Bayesian networks, the MDL metric, and MDLEP. In section 3, we describe our algorithm in detail. In sections 4 and 5, we report our experimental findings. We conclude the paper in section 6.

2 Learning BAYESIAN networks from data

2.1 Bayesian networks

A Bayesian network, G, has a directed acyclic graph (DAG) structure. Each node in the graph corresponds to a discrete random variable in the domain. An edge, $X \leftarrow Y$, on the graph, describes a parent and child relation in which X is the child and Y is the parent. All parents of X constitute the parent set of X which is denoted by Π_X. In addition to the graph, each node has a conditional probability tables (CPT) specifying the probability of each possible state of the node given each possible combination of states of its parent. If a node contains no parent, the table gives the marginal probabilities of the node [13].

Since Bayesian networks are founded on the idea of conditional independence, it is necessary to give a brief description here. Let U be the set of variables in the domain and let P be the joint probability distribution of U. Following Pearl's notation, a conditional independence (CI) relation is denoted by $I(X, Z, Y)$ where X, Y, and Z are disjoint subsets of variables in U. Such notation says that X and Y are conditionally independent given the *conditioning set*, Z. Formally, a CI relation is defined with:

$$P(x, y \mid z) = P(x \mid z) \quad \text{whenever} \quad P(y, z) > 0 \qquad (1)$$

where x, y,and z are any value assignments to the set of variables X, Y, and Z respectively. A CI relation is characterized by its *order*, which is the number of variables in the conditioning set Z.

As mentioned before, researchers treat the network learning problem in two very different ways. The first approach tries to construct a Bayesian network using dependency information obtained from the data. By assuming that P is faithful to a Bayesian network G [17], we could add or remove edges from G according to the discovered conditional independence relations. Given the sets of variables, X, Y, and Z, we could check the validity of $I(X, Z, Y)$ by performing statistical test, called CI test. The major problem of this approach is that it is difficult to know if two nodes are conditionally independent [17]. Furthermore, when a high-order CI relation is tested in a small data set, the test result may be unreliable [17]. The second approach makes use of a metric which evaluates the quality of a Bayesian network with respect to the given data. Such metric may be derived from information theory, Bayesian statistics, or Minimum Description Length principle (MDL). With the metric, the network learning problem becomes a search problem. Unfortunately, since the search space is huge, the search problem is difficult [6].

2.2 The MDL metric

The MDL metric [10] is derived from information theory and incorporates the Minimum Description Length principle. With the composition of the description length for network structure and the description length for data, the MDL metric tries to balance between model accuracy and model complexity. Hence, the best network needs to be both accurate and simple. Using the metric, a better network would have a smaller score. Similar to other metrics, the MDL score for a Bayesian network, G, is *decomposable* [6] and could be written as in equation 2. Let $U = \{N_1, \ldots, N_n\}$ be the set of nodes and let Π_{N_i} denotes the parent set of node N_i. The MDL score of the network is simply the summation of the MDL score of Π_{N_i} of every node N_i in the network.

$$\text{MDL}(G) = \sum_{N_i \in U} \text{MDL}(N_i,\ \Pi_{N_i}) \qquad (2)$$

2.3 MDLEP

Our previous work [18], called MDLEP, belongs to the score-and-search approach in which we use the MDL metric together with evolutionary programming (EP) for searching a good network structure. An individual in the search population is a candidate network structure. MDLEP uses simple, reversion, move, and knowledge-guided mutations to generate new individuals. When comparing MDLEP against another approach using GA [12], it is found that MDLEP generally outperforms its opponent.

3 Hybrid EP (HEP)

Although MDLEP outperforms its GA opponent, its efficiency can be enhanced by employing a number of strategies. First, a hybrid approach is introduced so that the knowledge from dependency tests is exploited during searching. Second, previous search results are reused through a new merge operator. Third, in contrast to MDLEP where repairing is needed, the formation of cycle is avoided altogether when producing new individuals.

Since a hybrid approach is adopted in Bayesian network learning, this approach is called HEP (hybrid EP). In the following subsections, the ideas will be discussed in detail.

3.1 A hybrid approach

In dependency analysis approach, CI test is typically used to check the validity of a conditional independence assertion $I(X, Z, Y)$ of any given two nodes X, Y and a conditioning set Z. Assume that the χ^2 test is employed, the assertion is modeled as the null hypothesis. A χ^2 test generates a p-value, ranges between 0 and 1, which shows the least level of significance for which the given data leads to the rejection of the null hypothesis. In effect, if the p-value is less than a predefined cutoff value, α, the hypothesis $I(X, Z, Y)$ is rejected. Otherwise, if the p-value is greater than or equal to α, the hypothesis could not be rejected and $I(X, Z, Y)$ is assumed to be valid. Consequently, this implies that the two nodes, X and Y, cannot have a direct edge between them. In other words, the edges $X \leftarrow Y$ and $X \rightarrow Y$ cannot exist in the resultant network.

With such observation, a hybrid framework for learning Bayesian networks is formulated which consists of two phases. In the first phase, low-order CI tests are conducted so that some edges could be removed. Only low-order CI tests are performed because their results are more reliable than higher order tests and the time complexity is bounded. In the second phase, a score-and-search approach is used together with the knowledge obtained previously. In particular, the search space is limited by excluding networks that contain the edges $X \leftarrow Y$ or $Y \rightarrow X$ for which $I(X, Z, Y)$ is assumed to be valid. Since the search space is reduced, the learning problem becomes easier and less time will be needed for finding the best network.

This idea could be applied readily in MDLEP. After obtaining the test results, all candidate networks having invalid edges are prevented from being generated.

Although such formulation can work fine, it must be emphasized that the choice of α has a critical impact. If improper α is used, in the worst case, either all edges are pruned away or all edges are retained. Hence, although it is possible to impose the restrictions from CI tests as *global* constraints, there is the risk of assuming our choice of α is appropriate.

As an alternative, a novel realization of the hybrid framework is developed in which a different α is used for each individual in the population. Thus, each individual has, besides the network structure, a cutoff value α which is also subjected to be evolved. As the evolutionary search proceeds, individual having an improper value of α will eventually be eliminated. In general, small value of α implies more constraints (less likely to reject an hypothesis) and results in a more restricted search space. Hence, if the value of α of an individual is too small which excludes some *important* edges, the individual will have a greater chance of being eliminated. On the other hand, if the value of α of an individual is too large, it is less likely to find the *right* edge (because there are many *wrong* alternatives) for its offspring. Consequently, the individual will also have a higher chance of being eliminated.

This idea is implemented in the first phase by storing the largest p-value returned by the CI tests for every possible conditioning set, Z (restricted to order-0 and all order-1 tests) in a matrix, Pv. In the second phase, for a given individual G_i in the population with associated cutoff value α_i, an edge $X \leftarrow Y$ cannot be added if Pv_{XY} is greater than α_i (i.e. $I(X, Z, Y)$ is assumed to be valid). The value of each α_i is randomly initialized in the beginning. In subsequent generations, an offspring will inherit the cutoff value from its parent with a possible increment or decrement by Δ_α.

3.2 The merge operator

In addition to the four mutation operators, a new operator called merge is introduced. Taking a parent network G_a and another network G_b as input, the merge operator attempts to produce a better network structure (in terms of MDL score) by modifying G_a with G_b. If no modification can be done, G_a is returned.

Let M_i^x denotes the MDL score of the parent set $\Pi_{N_i}^x$ of node $N_i \in U$ in the network G_x. Recalling that the MDL score is decomposable and a network is an agglomeration of Π_{N_i} (for $i = 1, \ldots, n$). Thus, given two input networks G_a and G_b, a better network, G_c, could be generated by selecting $\Pi_{N_i}^c$ from $\Pi_{N_i}^a$ or $\Pi_{N_i}^b$ so that (1) there is no cycle in G_c and (2) the sum $\sum_{N_i \in U} M_i^c$ is less than $\sum_{N_i \in U} M_i^a$. With such observation, the merge operator is devised and is the heuristics for finding a subset of nodes, $W \subset U$, with which $\Pi_{N_j}^a$ are replaced with $\Pi_{N_j}^b$ in G_a for every $N_j \in W$. Meanwhile, the replacement would not create cycles and has a MDL score smaller than that of G_a. The pseudo-code for the merge operator is presented in Table 1.

For the two input networks G_a and G_b, the merge procedure produces a node ordering by sorting $\delta_i = M_i^a - M_i^b$ in descending order. Since positive δ_i means that $\Pi_{N_i}^b$ is better

Procedure merge(G_a, G_b)

1. Find $\delta_i = M_i^a - M_i^b$ for every node $N_i \in U$.
2. Produce a node ordering L by sorting δ_i in descending order.
3. Set $W = \phi$.
4. While there are still nodes in L left unconsidered,
   - Get the next node, N_i, from L which is unconsidered.
   - Set $W' = \phi$.
   - Invoke the procedure `findSubset(`N_i`, `W'`)` which returns W' on completion.
   - Calculate the sum of δ_j for every node $N_j \in (W' - W)$.
   - If the sum is greater than zero
     - Mark every node $N_j \in W'$ in L as considered.
     - Replace $\Pi_{N_j}^a$ with $\Pi_{N_j}^b$ for every node $N_j \in (W' - W)$.
     - Set $W = W \cup W'$.

Table 1. Pseudo-code for the merge operator.

than $\Pi_{N_i}^a$, the procedure follows the ordering in considering the replacement of $\Pi_{N_i}^a$ with $\Pi_{N_i}^b$. Beginning with the first node, N_i, in the ordering, the merge procedure invokes the procedure `findSubset(`N_i`)` to find a subset of nodes W' such that by replacing $\Pi_{N_j}^a$ with $\Pi_{N_j}^b$ for every $N_j \in W'$ in G_a, the resultant graph is still acyclic.

After obtaining W', the merge procedure calculates the sum $\sum_{N_j \in (W'-W)} \delta_j$. If the sum is greater than zero, it replaces $\Pi_{N_j}^a$ with $\Pi_{N_j}^b$ in G_a for every $N_j \in (W' - W)$, removes W' from the ordering and then inserts W' into W. The procedure repeatedly examines the next node in the ordering until all nodes are considered.

Essentially, the merge operator increases the efficiency in several ways. Since the score of the composite network can be readily calculated, it is not necessary to invoke the procedure for MDL score evaluation which is time-consuming. Thus, the merge operator offers an economical way to create new structures. Furthermore, the operator improves the search efficiency by creating more good individuals in each generation. In our current implementation, the operator merges networks at the current population with dumped networks from the last generation. Thus, it reuses the search results obtained in previous generations.

3.3 Prevention of cycle formation

Since MDLEP consumes much time in repairing networks that contain cycles, HEP prevents cycle formation in all candidate networks to handle this problem. HEP maintains the *connectivity matrix* containing the count of directed paths between every pair of nodes. If $X \rightarrow \cdots \rightarrow Y$ exists in a network, HEP forbids adding the edge $X \leftarrow Y$ to the network. The matrix is updated when an edge is added or removed.

The algorithm of HEP is summarized in Table 2.

4 Comparing HEP with MDLEP

In our experiments, we compare HEP against MDLEP on a number of data sets generated from the ALARM Bayesian network, which appears in [18]. The data sets have respectively 1,000, 2,000, 5,000, and 10,000 cases. Since both algorithms are stochastic in nature, we have conducted 40 trials for each experiment. The programs are executed on the same Sun Ultra-5 workstation. For HEP, we set Δ_α to be 0.02. For both algorithms, the population size is 50 and the tournament size (q) is 7. We use 5000 generations as the common termination criterion and the maximum size of parent set is set to be 5. We compare the performance under five different aspects:

- average MDL score obtained, the smaller the better (AFS),
- average score of the first generation solution (AIS),
- average running time in seconds (AET),
- average generation that the best-so-far is found (ANG),
- average number of edges added, omitted, or reversed in compared to the original structure (ASD).

Table 3 provides a summary of the performance comparison between the two algorithms. The figures are average values of 40 trials. Numbers in parentheses are the standard deviations. The structural differences between the networks obtained by Bayesian Network Power Constructor (BNPC) [3] and the original networks are also presented. It can be observed that HEP performs better than BNPC, because the ASD values of HEP are smaller than those of BNPC in all data sets.

For all data sets, HEP could always find better or equally good network structures in terms of both MDL score (AFS) and structural difference (ASD). The difference is statistically significant at 0.05 level for all data sets. If we compare the ANG statistics, it is found that HEP uses much less generations to obtain the final solution (statistically significant at 0.05 level using a one-tailed t-test). Given that HEP and MDLEP essentially use the same formulation in searching, the experimental results readily suggest that HEP is more efficient as it uses fewer generations to obtain similar, or better, solutions. From the AET statistics, HEP uses much

CI Test Phase

1. For every pair of nodes (X, Y),
   - Perform order-0 and all order-1 CI tests.
   - Store the highest p-value in the matrix Pv.

Evolutionary Programming Search Phase

1. Set t, the generation count, to 0.
2. Initialize the value of m, the population size.
3. For each individual in the population Pop(t),
   - initialize the α value randomly.
   - refine the search space by checking the α value against the Pv matrix.
   - Inside the reduced search space, create a DAG randomly.
4. Each DAG in the population is evaluated using the MDL metric.
5. While t is less than the maximum number of generations,
   - select $m/2$ individuals from Pop(t), the rest are marked "NS" (not selected)
   - For each of the selected ones,
     - merge with a random pick from the dumped half in Pop$'(t-1)$.
     - If merge does not produce a new structure, mark the individual with "NS"
     - otherwise, regard the new structure as an offspring.
   - For each individual marked "NS",
     - produce an offspring by cloning.
     - alter the α value of the offspring by a possible increment or decrement of Δ_α.
     - refine the search space by checking the α value against the Pv matrix.
     - change the structure by performing a number of mutation operations. Note that cycle formation is prohibited.
   - The DAGs in Pop(t) and all new offspring are stored in the intermediate population Pop$'(t)$. The size of Pop$'(t)$ is 2*m.
   - Conduct a number of pairwise competitions over all DAGs in Pop$'(t)$. For each DAG G_i in the population, q other individuals are selected. The fitness of G_i is compared against the q individuals. The score of G_i is the number of individuals (out of q) that are worse than G_i.
   - Select the m highest score individuals from Pop$'(t)$ with ties broken randomly. The individuals are stored in Pop$(t+1)$.
   - increment t by 1
6. Return the individual that has the lowest MDL metric in any generation of a run as the output of the algorithm.

Table 2. Algorithm of HEP.

Size		AFS	AIS	AET	ANG	ASD
1000	HEP	17,880.56 (31.9)	24,323.5 (1,186.6)	204.75 (3.9)	817.6 (1,163.0)	11.15 (2.3)
	MDLEP	17,990.5 (73.1)	30,831.0 (795.6)	1,003.9 (70.8)	4,301.2 (654.3)	19.4 (4.2)
	BNPC	–	–	–	–	20
2000	HEP	33,777.8 (62.9)	44,199.45 (1,324.9)	225.63 (10.0)	1,410.78 (1,540.2)	9.05 (1.4)
	MDLEP	33,932.6 (215.8)	56,896.6 (1,259.5)	1,307.8 (125.1)	4,046.6 (634.1)	12.9 (4.9)
	BNPC	–	–	–	–	15
5000	HEP	81,004 (0.0)	102,310.02 (2,352.0)	290.3 (11.9)	448.57 (796.0)	6.05 (0.5)
	MDLEP	81,287.6 (419.9)	134,487.2 (1,836.0)	1,843.2 (359.0)	3,946.3 (651.2)	10.7 (4.9)
	BNPC	–	–	–	–	10
10000	HEP	158,498.5 (298.5)	199,210.75 (5,082.8)	384.77 (27.5)	970.42 (879.4)	4.53 (2.8)
	MDLEP	158,704.4 (513.1)	256,946.2 (3,843.7)	2,435.1 (350.1)	3,596.7 (720.0)	8.7 (5.1)
	BNPC	–	–	–	–	10

Table 3. Performance comparison between HEP and MDLEP

less time to finish than MDLEP under the same termination criterion.

If we compare the AIS statistics, it is clear that HEP could often have a better starting point than MDLEP. Apparently, this is also the benefit of the hybrid approach as we take CI test results into consideration rather than to initialize the population randomly.

5 Application in direct marketing

In this section, we investigate the feasibility of applying Bayesian networks on a real world data mining problem. The problem relates with direct marketing in which the objective is to predict buyers from a list of customers. Advertising campaign, which includes mailing of catalogs or brochure, is then targeted on the most promising prospects. Hence, if the prediction is accurate, it can help to enhance the *response rate* of the advertising campaign and increase the return of investment (ROI). The direct marketing problem requires ranking the customer list by the likelihood of purchase [19, 1]. Given that Bayesian networks estimate the posterior probability of an instance (a customer) belonging to a particular class (active or inactive respondents), they are particularly suitable for handling the direct marketing problem.

5.1 The direct marketing problem

Direct marketing concerns communication with prospects, so as to elicit response from them. In contrast to the mass marketing approach, direct marketing is targeted on a group of individuals that are potential buyers and are

likely to respond. In retrospect, direct marketing emerged because of the prevalence of mail ordering in the nineteenth century [14]. As technology advances, marketing is no longer restricted to mailing but includes a variety of media. Nevertheless, the most important issue in the business remains to be the maximization of the profitability, or ROI, of a marketing campaign.

In a typical scenario, we often have a huge list of customers. This list could be records of existing customers or data bought from *list brokers*. But among the huge list, there are usually few real buyers which amount to a few percents [2]. Since the budget of a campaign is limited, it is important to focus the effort on the most promising prospects so that the response rate could be improved.

Before computers became widely used, direct marketers often used simple heuristics to enhance the response rate. One straightforward approach is to use common sense to make the decision. In particular, we could match prospects by examining the demographics of the customers in the list. For example, in the life insurance industry, it is natural to target the advertising at those who are rich and aging. Another common approach to enhance the response rate is to conduct list testing by evaluating the response of samplings from the list. If a certain group of customers gives a high response rate, the actual campaign may be targeted on the customers similar to this group. A more systematic approach, which was developed in 1920s but is still being used today, is to differentiate potential buyers from non-buyers using the recency-frequency-monetary model (RFM) [14]. In essence, the profitability of a customer is estimated by three factors including the recency of buying, the frequency of buying, and the amount of money spent. Hence, only individuals that are profitable will be the targets of the campaign.

With the advancement of computing and database technology, people seek for computational approaches to assist in decision making. From the data set that contains demographic details of customers, the objective is to develop a *response model* and use the model to predict promising prospects. In certain sense, response models are similar to classifiers in the classification problem. However, unlike the classifier which makes a dichotomous decision (i.e. active or inactive respondents), the response model needs to score each customer in the data set with the likelihood of purchase. The customers are then ranked according to the score. A ranked list is desired because it allows decision makers to select the portion of customer list to roll out [19]. For instance, out of the 200,000 customers on the list, we might wish to send out catalogs or brochures to the most promising 30% of customers so that the advertising campaign is cost-effective (the 30% of the best customers to be mailed is referred to as the *depth-of-file*) [1]. Hence, one way to evaluate the response model is to look at its performance at different depth-of-file.

5.2 Experiment

Because Bayesian networks can estimate the probability of an object belonging to certain class(es), they are suitable to handle the direct marketing problem. By assuming the estimated probability to be equal to the likelihood of purchase, a Bayesian network is readily applicable to the direct marketing problem. Thus, it is interesting to evaluate the empirical performance of Bayesian network response models. Specifically, we compare the performance of the evolved Bayesian network models obtained by HEP and MDLEP, the logistic regression models, the naïve Bayesian classifier (NB) [5, 11], and the tree-augmented naïve Bayesian network classifier (TAN) [5]. NB simplifies the estimation of the joint probability distribution by assuming that each attribute is conditionally independent of others given the class variable. Although the assumption behind the naïve Bayesian classifier seems unrealistic [5], the classifier often exhibits surprisingly good and robust performance in many real-life problems [11]. TAN contains augmented edges which form a spanning tree. It is regarded as the state-of-the-art Bayesian network classifier [9].

For both HEP and MDLEP, the population size is 50, the maximum number of generations is 5000, and the tournament size (q) is 7. The maximum size of parent set is 5. For HEP, Δ_α is set to 0.02.

5.2.1 Experimental methodology

The response models are evaluated on a real-life direct marketing data set. It contains records of customers of a specialty catalog company, which mails catalogs to good customers on a regular basis. There is a total of 106,284 customers in the data set and each entry is described by 361 attributes. The response rate is 5.4%.

Typically in any data mining process, it is necessary to reduce the dimension of the data set by selecting the attributes that are considered relevant and necessary. Towards this feature selection process, there are many possible options. For instance, we could use either a *wrapper* or a *filter* selection process [16]. In a wrapper selection process, different combinations are iteratively tried and evaluated by building an actual model out of the selected attributes. In a filter selection process, a certain evaluation function, which is based on information theory or statistics, is defined to score a particular combination of attributes. Then, the final combination is obtained in a search process. In this experiment, we use a manual selection procedure. We have selected nine attributes, which are relevant to the prediction, out of the 361 attributes.

To compare the performance of different response models, we use decile analysis which estimates the enhance-

ments of the response rates for marketing at different depth-of-file. Essentially, the ranked list is equally divided into ten deciles. Customers in the first decile are the top ranked customers that are most likely to give response. On the other hand, customers in the tenth decile are ranked lowest. Then, a *gains table* is constructed to describe the performance of the response model. In a gains table, we collect various statistics at each decile, including [15]:

Percentage of Active: It is the percentage of active respondents in the decile.

Lift: It is calculated by dividing the percentage of active respondents by the response rate of the file. Intuitively, it estimates the enhancement by the response model in discriminating active respondents over a random approach for the current decile.

Cumulative Lift: It is calculated by dividing the cumulative percentage of active respondents by the response rate of the file. Intuitively, this evaluates how good the response model is for a given depth-of-file over a random approach. The measure provides an important estimate of the performance of the model.

5.2.2 Cross-validation results

To make a comparison concerning the robustness of the response models, we adopt a cross-validation approach for performance estimation. Specifically, we employ a 10-fold cross-validation where the ten folds are partitioned randomly. In Table 4, the experimental results for the Bayesian networks evolved by HEP (HEP models) are shown. We tabulate the statistics at each decile averaged over the ten runs. Numbers after the "$\pm$" sign are the standard deviations. Table 4 shows that the HEP models have cumulative lifts of 392.40, 287.30, and 226.70 in the first three deciles respectively, suggesting that by mailing to the top three deciles alone, the HEP models generate over twice as many respondents as a random mailing without a model.

To facilitate direct comparison, the cumulative lifts of different models are summarized in Table 5. In this table, the highest cumulative lift in each decile is highlighted in bold. The superscript $+$ represents the cumulative life of the HEP models is significant higher at 0.05 level than that of the corresponding models. The superscript $-$ represents the cumulative life of the HEP models is significant lower at 0.05 level than that of the corresponding models. Table 5 indicates that the logistic regression models have cumulative lifts of 342.27, 249.20, and 210.40 in the first three deciles respectively. The cumulative lifts of the HEP models are significantly higher than those of the logistic regression models at 0.05 level (p-values are 0.00002, 0.0, and 0.00002 respectively).

Table 5 shows that the Bayesian networks generated by MDLEP (MDLEP models) have cumulative lifts of 377.20, 287.40, and 220.40 in the first three deciles respectively. The cumulative lifts of the HEP models in the first and the third deciles are significantly higher than those of the MDLEP models at 0.05 level (p-values are 0.01189 and 0.00377 respectively). The TAN classifiers have cumulative lifts of 385.20, 278.00, and 220.00 in the first three deciles respectively. The cumulative lifts of the HEP models in the second and the third deciles are significantly higher than those of the TAN classifiers at 0.05 level (p-values are 0.00063 and 0.00008 respectively). The NB classifiers have cumulative lifts of 378.10, 274.40, and 222.20 in the first three deciles respectively. The cumulative lifts of the HEP models in the first three deciles are significantly higher than those of the NB classifiers at 0.05 level (p-values are 0.00301, 0.00089, and 0.02793 respectively). Overall, the HEP models perform significantly better than the other models in predicting consumer response to direct mailing promotions.

Decile	Percent Actives	Lift	Cum. Lift
1	21.21% $\pm$ 1.34%	392.4 $\pm$ 19.36	392.4 $\pm$ 19.36
2	9.88% $\pm$ 0.87%	182.3 $\pm$ 12.48	287.3 $\pm$ 6.18
3	5.69% $\pm$ 0.64%	105.1 $\pm$ 12.60	226.7 $\pm$ 6.15
4	4.84% $\pm$ 0.63%	89.3 $\pm$ 13.37	192.2 $\pm$ 3.94
5	3.47% $\pm$ 0.78%	63.8 $\pm$ 13.80	166.6 $\pm$ 3.17
6	2.96% $\pm$ 0.72%	54.1 $\pm$ 12.91	147.7 $\pm$ 1.34
7	2.07% $\pm$ 0.51%	37.8 $\pm$ 8.69	132.2 $\pm$ 0.92
8	1.58% $\pm$ 0.27%	28.6 $\pm$ 4.65	119.2 $\pm$ 1.03
9	1.38% $\pm$ 0.36%	25.0 $\pm$ 7.06	108.8 $\pm$ 0.42
10	0.92% $\pm$ 0.28%	16.6 $\pm$ 5.13	100.0 $\pm$ 0.00

Table 4. Results of the HEP models.

Decile	HEP	Logistic regression	MDLEP	TAN	NB
0	**392.4** (19.36)	342.7^{+} (12.82)	377.2^{+} (20.04)	385.2 (18.73)	378.1^{+} (14.18)
1	287.3 (6.18)	249.2^{+} (7.96)	**287.4** (6.90)	278.0^{+} (8.89)	274.4^{+} (8.91)
2	**226.7** (6.15)	210.4^{+} (5.10)	220.4^{+} (2.84)	220.0^{+} (4.81)	222.2^{+} (4.37)
3	192.2 (3.94)	186.7^{+} (2.58)	**200.0**$^{-}$ (4.57)	187.4^{+} (3.78)	187.6^{+} (3.53)
4	**166.6** (3.17)	163.1^{+} (1.45)	160.6^{+} (3.37)	164.0^{+} (2.91)	162.5^{+} (1.90)
5	147.7 (1.34)	144.9^{+} (1.52)	**150.1**$^{-}$ (2.47)	145.1^{+} (2.02)	145.3^{+} (1.64)
6	**132.2** (0.92)	130.7^{+} (1.49)	128.7^{+} (2.11)	130.9^{+} (0.88)	130.5^{+} (1.18)
7	**119.2** (1.03)	118.4 (1.35)	118.7 (0.95)	118.4^{+} (0.97)	118.9 (1.10)
8	108.8 (0.42)	108.2^{+} (0.42)	**110.9**$^{-}$ (0.32)	108.1^{+} (0.74)	108.1^{+} (0.57)
9	100.0 (0.00)	100.0 (0.00)	100.0 (0.00)	100.0 (0.00)	100.0 (0.00)

Table 5. Cumulative lifts of different models.

The average execution time and the corresponding standard deviations for different methods are summarized in

Table 6. Although HEP is slower than logistic regression, TAN, and NB, it is much faster than MDLEP. Moreover, HEP is able to learn Bayesian networks from a large database in one minute. Thus, it can be used in real-life data mining applications.

	HEP	Logistic regression	MDLEP	TAN	NB
Average (sec.)	54.676	6.355	1999.04	20.101	19.598
Std.	3.838	0.5035	324.695	1.1503	0.9727

Table 6. The execution time for different methods.

Since an advertising campaign often involves huge investment, a response model which can categorize more prospects into the target list is valuable as it will enhance the response rate. From the experimental results, it seems that the HEP models are more desirable than the other models.

6 Conclusion

In this paper, we have described a new algorithm, HEP, for learning Bayesian networks efficiently. We have applied HEP to a data set of direct marketing and compared the Bayesian networks obtained by HEP and the models generated by other methods. From the experimental results, the HEP models predict more accurately than the other models. This study shows that HEP can potentially become a powerful and efficient data mining tool for direct marketing problems.

In our current implementation, we change the cutoff value of an offspring by arbitrarily increasing or decreasing a fixed value of Δ_α from the parent's value. However, it is also possible to use an adaptive mutation strategy such that the Δ_α will become smaller as time proceeds. In effect, the search space is gradually stabilized which may lead to a further speed up. In future, we will explore this and other alternatives that are worth investigating.

Acknowledgments

This research was partially supported by the RGC Earmarked Grant LU 3012/01E.

References

[1] S. Bhattacharyya. Direct marketing response models using genetic algorithms. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, pages 144–148, 1998.

[2] P. Cabena, P. Hadjinian, R. Stadler, J. Verhees, and A. Zansi. *Discovering Data Mining: From Concept to Implementation*. Prentice-Hall Inc., 1997.

[3] J. Cheng, R. Greiner, J. Kelly, D. Bell, and W. Liu. Learning Bayesian network from data: An information-theory based approached. *Artificial Intelligence*, 137:43–90, 2002.

[4] U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors. *Advances in Knowledge Discovery and Data Mining*. AAAI Press, 1996.

[5] N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29:131–163, 1997.

[6] D. Heckerman. A tutorial on learning Bayesian networks. Technical report, Microsoft Research, Advanced Technology Division, March 1995.

[7] E. Herskovits and G. Cooper. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9(4):309–347, 1992.

[8] F. V. Jensen. *An Introduction to Bayesian Network*. University of College London Press, 1996.

[9] E. J. Keogh and M. J. Pazzani. Learning augmented Bayesian classifiers: A comparison of distribution-based and classification-based approaches. In D. Heckerman and J. Whittaker, editors, *Proceedings of the Seventh International Workshop on AI and Statistics*, pages 225–230, Fort Lauderdale, Florida, January 1999. Morgan Kaufmann.

[10] W. Lam and F. Bacchus. Learning Bayesian belief networks-an approach based on the MDL principle. *Computational Intelligence*, 10(4):269–293, 1994.

[11] P. Langley and S. Sage. Induction of selective Bayesian classifier. In R. L. de Mantaras and D. Poole, editors, *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, Seattle, Washington, July 1994. Morgan Kaufmann.

[12] P. Larrañaga, M. Poza, Y. Yurramendi, R. Murga, and C. Kuijpers. Structural learning of Bayesian network by genetic algorithms: A performance analysis of control parameters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(9):912–926, September 1996.

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

[14] L. A. Petrison, R. C. Blattberg, and P. Wang. Database marketing: Past present, and future. *Journal of Direct Marketing*, 11(4):109–125, 1997.

[15] O. P. Rud. *Data Mining Cookbook: modeling data for marketing, risk and customer relationship management*. Wiley, New York, 2001.

[16] M. Singh. *Learning Bayesian Networks for Solving Real-World Problems*. PhD thesis, University of Pennsylvania, 1998.

[17] P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction, and Search*. MIT Press, MA, second edition, 2000.

[18] M. L. Wong, W. Lam, and K. S. Leung. Using evolutionary programming and minimum description length principle for data mining of Bayesian networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(2):174–178, February 1999.

[19] J. Zahavi and N. Levin. Issues and problems in applying neural computing to target marketing. *Journal of Direct Marketing*, 11(4):63–75, 1997.