

---

# A Hybrid Data Mining Approach to Discover Bayesian Networks 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

Given the explosive growth of data collected from current business environment, data mining can potentially discover new knowledge to improve managerial decision making. We propose a novel data mining approach that employs evolutionary programming to discover knowledge represented in Bayesian networks and apply the approach to marketing data. There are two different approaches to the network learning problem. The first one uses dependency analysis, while the second approach searches good network structures according to a metric. Unfortunately, the two approaches both have their own drawbacks. Thus, we propose a novel hybrid of the two approaches. With this new idea, we endeavor to improve upon our previous work, MDLEP, which uses evolutionary programming for network learning. We also introduce a new operator to further enhance the search efficiency. We conduct a number of experiments and compare the hybrid approach with MDLEP. The empirical results illustrate that the approach improves over MDLEP.

## 1 INTRODUCTION

Conventional marketing 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. Through data mining, marketing 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 to discover knowledge represented in Bayesian Networks and apply the approach to 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 [1]. 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 [2].

Typically, a Bayesian network is constructed by eliciting knowledge from domain experts. To reduce imprecision due to subjective judgements, some researchers propose methods to construct Bayesian networks from collected data. In the literature, there are two main approaches to this network learning problem [3]. The first one is the dependency analysis approach [4, 3]. 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 [5, 6, 7], uses a metric to evaluate a candidate network structure. With the metric, a search algorithm is employed to find a network structure having 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 should be performed. Moreover, some test results may be inaccurate [4]. For the latter approach, the search space is huge. Therefore, some algorithms [5] adopt greedy search heuristics which may easily make

the algorithms get stuck in a local optimum [7].

In this work, a hybrid framework 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, we call our new approach HEP (hybrid MDLEP). We have conducted a number of experiments and compared HEP with MDLEP. The empirical results illustrate that HEP improves over MDLEP. Moreover, it is found that HEP executes much faster than MDLEP which is very important for real-world applications.

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  $\Pi_X$ . 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 [1].

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:

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

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  $G$  [4], 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 dependent [4]. Furthermore, when a high-order CI relation is tested in a small data set, the test result may be unreliable [4]. 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 [7].

### 2.2 THE MDL METRIC

The MDL metric [6] 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* [7] and could be written as in equation 2. Let  $U = \{N_1, \dots, N_n\}$  be the set of nodes and let  $\Pi_{N_i}$  denotes the parent set of node  $N_i$ . The MDL score of the network is simply the summation of the MDL score of  $\Pi_{N_i}$  of every node  $N_i$  in the network.

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

### 2.3 MDLEP

Our previous work [8], 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. An outline of the algorithm is presented as follows:

1. Set  $t$ , the generation count, to 0.
2. Create an initial population,  $\text{Pop}(t)$  of  $m$  random DAGs ( $m$  is the population size.)
3. Each DAG in the population is evaluated using the MDL metric.
4. While  $t$  is less than the maximum number of generations,
  - Each DAG in  $\text{Pop}(t)$  produces one offspring by performing a number of mutation operations. Four different mutation operators, simple mutation, reverse mutation, move mutation, and knowledge-guided mutation are used. If there is cycle, a randomly picked edge in the cycle is removed.
  - The DAGs in  $\text{Pop}(t)$  and all new offspring are stored in the intermediate population  $\text{Pop}'(t)$ . The size of  $\text{Pop}'(t)$  is  $2*m$ .
  - Conduct a number of pairwise competitions over all DAGs in  $\text{Pop}'(t)$ . For each  $G_i$  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  $\text{Pop}'(t)$  with ties broken randomly. The individuals are stored in  $\text{Pop}(t+1)$ .
  - increment  $t$  by 1.
5. Return the individual that has the lowest MDL metric in any generation of the run as the output of the algorithm.

When comparing MDLEP against another approach using GA [9], it is found that MDLEP generally outperforms its opponent.

### 3 HYBRID MDLEP (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 MDLEP). 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  $\chi^2$  test is employed, the assertion is modeled as the null hypothesis. A  $\chi^2$  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,  $\alpha$ , the hypothesis  $I(X, Z, Y)$  is rejected. Otherwise, if the  $p$ -value is greater than or equal to  $\alpha$ , 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  $\alpha$  has a critical impact. If improper  $\alpha$  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  $\alpha$  is appropriate.

As an alternative, a novel realization of the hybrid framework is developed in which a different  $\alpha$  is used for each individual in the population. Thus, each individual has, besides the network structure, a cutoff

value  $\alpha$  which is also subjected to be evolved. As the evolutionary search proceeds, individual having an improper value of  $\alpha$  will eventually be eliminated. In general, small value of  $\alpha$  implies more constraints (less likely to reject an hypothesis) and results in a more restricted search space. Hence, if the value of  $\alpha$  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  $\alpha$  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  $\alpha_i$ , an edge  $X \leftarrow Y$  cannot be added if  $Pv_{XY}$  is greater than  $\alpha_i$  (i.e.  $I(X, Z, Y)$  is assumed to be valid). The value of each  $\alpha_i$  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  $\Delta_\alpha$ .

### 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  $\Pi_{N_i}$  (for  $i = 1, \dots, n$ ). 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  $\sum_{N_i \in U} M_i^a$  or  $\sum_{N_i \in U} M_i^b$ . With such observation, the merge operator is devised which is an heuristic 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  $G_a$ .

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  $\delta_i$  means that  $\Pi_{N_i}^b$  is better than  $\Pi_{N_i}^a$ , the procedure follows the ordering in considering the replacement of  $\Pi_{N_i}^a$

#### 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  $\delta_i$  in descending order.
3. While there are nodes in  $L$  that have not been considered,
  - Get the next node,  $N_i$ , from  $L$  which is unconsidered.
  - Invoke `findSubset( $N_i$ )` which returns  $W'$ .
  - Sum  $\delta_j$  for every node  $N_j \in W'$ .
  - If the sum is positive, mark every node  $N_j \in W'$  in  $L$  as considered. Insert  $W'$  to  $W$ .
4. Replace  $\Pi_{N_j}^a$  with  $\Pi_{N_j}^b$  for every  $N_j \in W$ .

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  $\sum_{N_j \in W'} \delta_j$ . If the sum is positive, it inserts  $W'$  into  $W$  and removes  $W'$  from the ordering. The procedure repeatedly examines the next node in the ordering until all nodes are considered. Finally, the procedure replaces  $\Pi_{N_j}^a$  with  $\Pi_{N_j}^b$  in  $G_a$  for every  $N_j \in W$ .

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  $X \rightarrow \dots \rightarrow Y$  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.

### 3.4 THE ALGORITHM

The algorithm of HEP is presented as follows:

#### CI test Phase

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  $\text{Pop}(t)$ ,
  - initialize the  $\alpha$  value randomly.
  - refine the search space by checking the  $\alpha$  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  $\text{Pop}(t)$ , the rest are marked “NS” (not selected)
  - For each of the selected ones,
    - merge with a random pick from the dumped half in  $\text{Pop}'(t-1)$ .
    - 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  $\alpha$  value of the offspring by a possible increment or decrement of  $\Delta_\alpha$ .
    - refine the search space by checking the  $\alpha$  value against the  $Pv$  matrix.
    - change the structure by performing a number of mutation operations. Note that cycle formation is prohibited.
  - The DAGs in  $\text{Pop}(t)$  and all new offspring are stored in the intermediate population  $\text{Pop}'(t)$ . The size of  $\text{Pop}'(t)$  is  $2*m$ .

- Conduct a number of pairwise competitions over all DAGs in  $\text{Pop}'(t)$ . 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  $\text{Pop}'(t)$  with ties broken randomly. The individuals are stored in  $\text{Pop}(t+1)$ .
- 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.

## 4 COMPARISON WITH MDLEP

In our experiments, we compare HEP against MDLEP on a number of data sets. We use the data sets that are generated from two benchmark networks, ALARM and PRINTD which also appear in [8]. There are data sets of sizes 1,000, 2,000, 5,000, and 10,000 for ALARM and one data set of 5,000 cases for PRINTD. 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  $\Delta_\alpha$  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 five. We compare the performance under five different aspects:

- average MDL score obtained, the smaller the better (Score),
- average running time in seconds (Time),
- average score of the first generation solution (I-Score),
- 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 1 provides a summary of the performance comparison between the two algorithms. The MDL score of the original network is also included (just below the name of each data set) as reference. Numbers in parentheses are the standard deviations.

For all data sets, HEP could always find better or equally good network structures in terms of both MDL

Table 1 Performance comparison between the two algorithms over different data sets

Data Set		Score	Time	I-Score	ANG	ASD
ALARM 1000 (18533.5)	HEP	17871.7 (37.2)	212.1 (12.1)	24542.5 (1103.6)	750.1 (1101.4)	12.0 (1.7)
	MDLEP	17974.4 (87.1)	1115.4 (70.2)	30833.6 (786.3)	4353.3 (666.6)	18.7 (4.0)
ALARM 2000 (34287.9)	HEP	33800.5 (86.8)	273.6 (33.5)	44188.5 (1179.2)	1012.7 (1409.2)	8.5 (1.5)
	MDLEP	34018.3 (205.6)	1608.5 (196.9)	57138.3 (1228.6)	4163.8 (762.9)	13.5 (3.9)
ALARM 5000 (81233.4)	HEP	81004.0 (0.0)	491.8 (116.7)	102002.0 (2254.3)	421.1 (737.6)	7.1 (0.4)
	MDLEP	81237.9 (432.4)	3224.3 (449.8)	133901.7 (1968.4)	4016.9 (677.6)	11.1 (4.1)
ALARM 10000 (158497.0)	HEP	158425.0 (30.2)	942.5 (358.8)	197090.0 (5398.9)	932.0 (995.9)	3.6 (0.8)
	MDLEP	158677.7 (394.8)	6443.0 (540.9)	257057.5 (4562.1)	3653.2 (800.5)	7.6 (3.7)
PRINTD 5000 (106541.6)	HEP	106541.6 (0.0)	224.5 (43.4)	111113.0 (447.0)	40.0 (8.8)	0.0 (0.0)
	MDLEP	106541.6 (0.0)	1692.2 (28.1)	116008.2 (509.1)	510.9 (87.4)	0.0 (0.0)

score and structural difference. Hence, under the same termination criterion, it suggests that HEP is more efficient than MDLEP. Moreover, our approach consumes much less time than MDLEP. In some case, the saving is so great that HEP has nearly an eight-fold speedup. Hence, HEP is more favorable to MDLEP for real world practice. Since we have adopted the hybrid framework for creating the initial population, HEP usually has a better starting point than MDLEP as reflected by I-score.

## 5 APPLICATION IN DATA MINING

In this section, HEP is applied to a data mining problem in direct marketing. The objective of the problem is to predict potential prospects from the buying records of previous customers. Advertising campaign, which includes mailing of catalogs or brochures, is then targeted on the group of potential 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 is similar to the classification problem. However, rather than producing a clear cut between potential buyers from non-buyers, the direct marketing problem requires ranking the customer database according to the likelihood of purchase [10]. Since Bayesian networks can estimate the posterior probability of an instance (a customer) belonging to a particular class (buyer or non-buyer), they are particularly suitable for handling the direct marketing problem.

### 5.1 THE DIRECT MARKETING PROBLEM

Direct marketing concerns communication with prospects hoping to eliciting response from them. In contrast to the mass marketing approach, direct marketing is often targeted on a group of individuals that are potential buyers and are likely to give a response.

In a typical scenario, we often have a huge list of potential prospects. 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 only one or two percents [10]. Since the budget of a campaign is limited, it is important to target the effort on potential buyers so that the response rate could be improved.

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 potential buyers. In certain sense, response models are similar to classifiers in the classification problem. However, unlike a classifier which makes a dichotomous decision (i.e. active or non-active), a 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 customers to roll out to [11]. For instance, out of the 200,000 customers on the list, we might wish to send out catalogs or brochures to the most promising 20% of customers so that the advertising campaign

is cost-effective. Hence, one way to evaluate a response model is to look at its performance at different *depth-of-file*. In the literature, there are various approaches proposed for building response models [12, 13, 14].

## 5.2 EXPERIMENT

Because Bayesian networks can estimate the probability of belonging to certain class(es), they are also 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 models evolved by HEP with those obtained by MDLEP.

### 5.2.1 Experiment Methodology

The response models are evaluated on a real-life data set. The data set 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 278 attributes.

Typical 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* selection process or a *filter* selection process [15]. 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, 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 have manually selected nine attributes, that are relevant to prediction, out of the 278 attributes.

To compare the performance of different response models, we use decile analysis which estimates the enhancement of the response rate 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 tabulate various statistics at each decile, including [10]:

**Percentage of Active** The actual percentage of ac-

tive respondents in the decile.

**Lift** Lift 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** The cumulative lift is calculated by dividing the cumulative percentage of active respondents by the response rate. Intuitively, this evaluates how good the response model is for a given depth-of-file over a random approach.

### 5.2.2 Cross-Validation Result

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 Tables 2 and 3, the experimental results for the models evolved by MDLEP and those obtained by HEP are shown respectively. We collect the statistics on the percentage of active response, lift, and cumulative lift at each decile averaged over the ten runs. The numbers after the “±” sign are the standard deviations. Table 2 indicates the first two deciles have cumulative lifts of 377.2 and 287.4 respectively, suggesting that by mailing to the top two deciles alone, the MDLEP models generate over twice as many respondents as a random mailing without a model. However, the lift in the third decile declines sharply to 86.1, which is lower than the next decile (139.4). The lifts in the fifth, seventh, and tenth deciles decrease sharply to 3.2, 0.7, and 0.4, respectively. This phenomenon suggests instability in the models. The results in Table 3 show that the HEP models have a cumulative lift of 392.4 in the top decile, higher than that of the MDLEP models. The HEP models are more stable because there is no sudden drop in lifts in all deciles.

Decile	Percent Actives	Lift	Cum. Lift
1	20.39% ± 1.14%	377.2 ± 20.04	377.2 ± 20.04
2	10.71% ± 1.64%	197.4 ± 26.17	287.4 ± 6.90
3	4.67% ± 0.75%	86.1 ± 15.04	220.4 ± 2.84
4	7.53% ± 0.60%	139.4 ± 14.80	200.0 ± 4.57
5	0.18% ± 0.38%	3.2 ± 6.89	160.6 ± 3.37
6	5.31% ± 0.75%	97.7 ± 13.33	150.1 ± 2.47
7	0.04% ± 0.12%	0.7 ± 2.21	128.7 ± 2.11
8	2.81% ± 0.51%	51.2 ± 7.48	118.7 ± 0.95
9	2.34% ± 0.47%	42.9 ± 7.55	110.9 ± 0.32
10	0.03% ± 0.09%	0.4 ± 1.26	100.0 ± 0.00
Total	5.40% ± 0.22%		

Table 2: Result of the MDLEP model.

Since an advertising campaign often involves huge investment, a response model which can categorize more potential prospects into the target list, which amount

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
Total	5.40% $\pm$ 0.22%		

Table 3: Result of the HEP model.

to the first few deciles, is useful as it will enhance the response rate. From the experimental results, it seems that the HEP models are more desirable than the MDLEP models.

## 6 CONCLUSION

In this paper, we have reported a hybrid framework for learning Bayesian networks and have incorporated the ideas into HEP. In addition, we have also devised a new operator and modified a weakness of MDLEP. As experimental results suggest, our new approach (HEP) is much more efficient than MDLEP. Most importantly, we note that the new approach provides a tremendous speedup.

We apply HEP to a data set of marketing and compare the models obtained by HEP and those produced by MDLEP. From the experimental results, the HEP models predict more accurately than the MDLEP models.

In our current implementation, we change the cutoff value of an offspring by arbitrarily increasing or decreasing a fixed value of  $\Delta_\alpha$  from the parent's value. However, it is also possible to use an adaptive mutation strategy such that the  $\Delta_\alpha$  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] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.
- [2] F. V. Jensen, *An Introduction to Bayesian Network*. University of College London Press, 1996.
- [3] J. Cheng, D. A. Bell, and W. Liu, "Learning Bayesian networks from data: An efficient approach based on information theory," in *Proceedings of the Sixth ACM International Conference on Information and Knowledge Management*, (Las Vegas, Nevada), pp. 325–331, ACM, November 1997.
- [4] P. Spirtes, C. Glymour, and R. Scheines, *Causation, Prediction, and Search*. MA: MIT Press, second ed., 2000.
- [5] E. Herskovits and G. Cooper, "A Bayesian method for the induction of probabilistic networks from data," *Machine Learning*, vol. 9, no. 4, pp. 309–347, 1992.
- [6] W. Lam and F. Bacchus, "Learning Bayesian belief networks—an approach based on the MDL principle," *Computational Intelligence*, vol. 10, no. 4, pp. 269–293, 1994.
- [7] D. Heckerman, "A tutorial on learning Bayesian networks," tech. rep., Microsoft Research, Advanced Technology Division, March 1995.
- [8] 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*, vol. 21, pp. 174–178, February 1999.
- [9] P. Larrañaga, C. Kuijpers, R. Mura, and Y. Yurramendi, "Structural learning of Bayesian network by genetic algorithms: A performance analysis of control parameters," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, pp. 912–926, September 1996.
- [10] O. P. Rud, *Data Mining Cookbook: modeling data for marketing, risk and customer relationship management*. New York: Wiley, 2001.
- [11] J. Zahavi and N. Levin, "Issues and problems in applying neural computing to target marketing," *Journal of Direct Marketing*, vol. 11, no. 4, pp. 63–75, 1997.
- [12] L. A. Petrison, R. C. Blattberg, and P. Wang, "Database marketing: Past present, and future," *Journal of Direct Marketing*, vol. 11, no. 4, pp. 109–125, 1997.
- [13] S. Bhattacharyya, "Evolutionary algorithms in data mining: Multi-objective performance modeling for direct marketing," in *Proceedings of the Sixth International Conference on Knowledge Discovery and Data Mining*, pp. 465–473, August 2000.
- [14] C. X. Ling and C. Li, "Data mining for direct marketing: Problems and solutions," in *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, pp. 73–79, 1998.
- [15] M. Singh, *Learning Bayesian Networks for Solving Real-World Problems*. PhD thesis, University of Pennsylvania, 1998.