A-HEP: Adaptive Hybrid Evolutionary Programming for Learning Bayesian Networks

Kit-Ying Lee[1], Man-Leung Wong[2], Yong Liang[1], Kwong-Sak Leung, and Kin-Hong Lee

[1] Department of Computer Science & Engineering,
The Chinese University of Hong Kong, Shatin, N.T., Hong Kong
[2] Department of Information Systems,
Lingnan University, Tuen Mun, N.T., Hong Kong
{kylee, yliang, ksleung, khlee}@cse.cuhk.edu.hk,
mlwong@ln.edu.hk

Abstract. This paper describes an optimized algorithm for learning Bayesian Network structure by using adaptive population sized evolutionary programming. Bayesian network (BN) is a popular knowledge discovery model which can represent the causal relationship of different events or attributes with uncertainty. Learning the structure solely by dependency analysis or search-and-score approach is not effective. The hybrid algorithm on evolutionary programming, HEP, has been shown to be effective and efficient to solve this learning problem [8]. By introducing the concept of adjusting the population size according to the individuals' dissimilarity, HEP is further optimized on the execution time with comparable performance. The empirical results illustrate that the optimized algorithm has reduced the running time by half.

1 Introduction

Bayesian network (BN) is a popular knowledge representation for machine learning and data mining. Owing to its ability of representing causality and uncertainty, it is widely used in various domains. With the use of Bayesian networks, many medical and operational diagnostic and prediction systems can be developed. Typically, a Bayesian network can be constructed by eliciting knowledge from domain experts. To reduce imprecision due to subjective judgments, many algorithms are developed for learning Bayesian networks from collected data and past observations in the domain.

There are two major approaches to this network learning problem [1]. The first one is the dependency analysis approach. Since a Bayesian network describes conditional independence, we can make use of dependency results to construct a Bayesian network that conforms to our findings. The drawback of this approach is the exponential number of dependency tests required. The second approach is the score-and-search approach [2][4][3][8][7]. A metric is used to evaluate different Bayesian networks, thus the learning problem becomes a search problem.

However, the algorithms applying this approach may stuck in a local optimum [2].

In our previous work, a hybrid approach is used for learning Bayesian network structures by evolutionary programming (HEP) [8]. HEP searches the network structure with the help of the statistical dependency information. It is shown to be effective and efficient in this learning problem. On the other hand, we have designed the adaptive elitist-population search method (AEGA) that locates all optima of multimodal problems [5]. In this paper, A-HEP is described that extends HEP by adopting the dynamic population size concept of AEGA. Since the running time of evolutionary algorithms depend on the population size, A-HEP can increase and decrease the population size adaptively according to the dissimilarity of individuals. Once the algorithm converges to a certain degree, the population size decreases and computation is also reduced. Compared with original HEP, A-HEP can reduce the execution time significantly.

This paper is organized as follows. In section 2, we present the background of Bayesian networks, HEP and AEGA. In section 3, we describe our algorithm in detail. In sections 4 and 5, we report our experimental findings and conclude our work.

2 Background

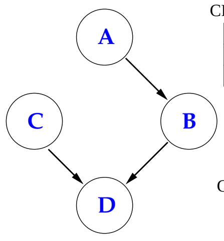
In this section, the general idea of Bayesian networks and HEP are described. Related work on AEGA are also introduced in the last part of this section.

2.1 Bayesian Networks

A Bayesian network, G, has a directed acyclic graph (DAG) structure. As shown in Figure 1, 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 table (CPT) specifying the probability of each possible state of the node given each possible combination of states of its parents. If a node contains no parent, the table gives the marginal probabilities of the node [6].

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 P be the joint probability distribution of U. Following Pearl's notation [6], 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 as [6]:

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



CPT of B

	A='F'	A='T'
B='F'	0.23	0.6
B='T'	0.77	0.4

CPT of D

	B='F', C='F'	B='F', C='T'	B='T', C='F'	B='T', C='T'
D='F'	0.65	0.54	0.78	0.01
D='T'	0.35	0.46	0.22	0.99

Fig. 1. A Bayesian network example.

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 simply the number of variables in the conditioning set Z.

By definition, a Bayesian network encodes the joint probability distribution of the domain variables, $U = \{N_1, \ldots, N_n\}$:

$$P(N_1, \ldots, N_n) = \prod_i P(N_i \mid \Pi_{N_i}). \tag{2}$$

2.2 Hybrid EP (HEP)

As mentioned before, researchers treat the network learning problem in two very different ways. They are called the dependency analysis and the search-and-scoring approaches respectively. Both approaches have their drawbacks. Hybrid EP (HEP), an extension of MDLEP [7], was designed to incorporate the dependency information into the searching process. The combination of the two approaches achieves better efficiency and improves the solution quality in a smaller number of generations [8].

The evolutionary part of HEP is similar to MDLEP, except that HEP has an additional CI test Phase just before the EP (Evolutionary Programming) Search Phase. For every pair of nodes, the order-0 and order-1 CI tests are done to find the *p-value* which indicates the dependency level between them. The search space is refined in each generation by checking the *alpha value*, the dependency threshold, against the p-value matrix.

Mutation operators used in HEP include simple mutation, reversion mutation, move mutation and knowledge-guided mutation. The last operator is similar to single mutation except that an edge is selected by comparing the corresponding MDL score of the connecting nodes. Edges with larger MDL scores tend to be removed while edges with smaller MDL scores tend to be added. In the HEP framework, a new operator *merge* is also introduced for better evolution. Taking a parent network G_a and another network G_b as input, the merge operator attempts to produce a better network by modifying G_a with G_b. If no modification can be done, G_a is returned. The use of merge operator is shown to improve the effectiveness and the efficiency of HEP. Algorithm 1 is the outline.

Algorithm 1 Algorithm of HEP

```
CI Test Phase
  for each pair of nodes $(X, Y)$ do
    Perform order-0 and all order-1 CI tests;
    Store the highest $p$-vale in the matrix $P_v$;
  end for
Evolutionary Programming Search Space
  Set $t$, the generation count, to 0;
  Initialize and evaluate the population with size $m$;
  for each individual $G_i$ in the population $Pop(t)$ do
    Initialize the $\alpha$ value randomly;
    Refine the search space by checking the $\alpha$ value against the $P_v$ matrix;
    Create a DAG randomly in the reduced search space;
  end for
  Each DAG in the population is evaluated using the MDL metric;
  while $t$ is less than the maximum number of generations do
    Randomly select $m/2$ individuals from $Pop(t)$, the rest are marked $NS$;
    for each of the selected ones do
      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;
    end for
    for each individual marked $NS$ do
      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 $P_v$ matrix;
      Change the structure by performing a number of mutation operations;
    end for
    The DAGs in $Pop(t)$ and all new offspring are stored in the intermediate popula-
    tion $Pop'(t)$ with size $2 * m$;
    Conduct pairwise competitions over all DAGs in $Pop'(t)$. For each $G_i$ in the popu-
    lation, its fitness is compared against $q$ individuals. The score of $G_i$ is the number
    of individuals (out of $q$) that are worse than $G_i$;
    Store the $m$ highest score individuals from $Pop'(t)$ with ties broken randomly in
    $Pop(t+1)$;
    Increment $t$ by 1;
  end while
  Return the final structure with the lowest MDL score in any generation of a run.
```

2.3 Adaptive Elitist-population Based Genetic Algorithm (AEGA)

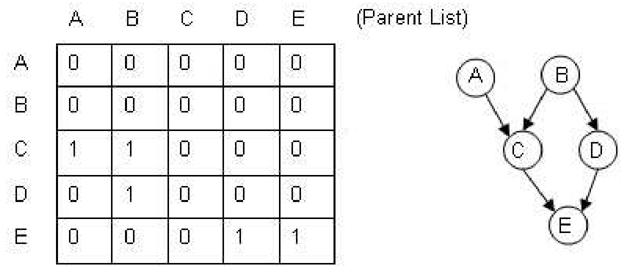
AEGA is a new technique used to solve multimodal function maximization problems. Elitist individuals are defined as the best ones on different peaks. With the help of elitist operators, the diversity and of population can be maintained and even improved by adjusting the population size according to the dissimilarity and directions of individuals in its population. Eventually, the population can exploit all optima of multimodal problems in parallel based on elitism [5]. The adaptive population size concept is adopted in A-HEP for optimization.

3 Proposed A-HEP Algorithm

The principle used for improving HEP is the adaptive population size concept according to the dissimilarity of individuals. Similar to other evolutionary algorithms, the population of HEP converges into one or several solutions with best fitness at the end of evolution. In the later part of evolution, most individuals are similar or exactly the same. Computation time can be reduced if these redundant individuals are removed from the population in the later generations. On the other hand, the diversity of the population are also important for searching the optimal solution, especially at the early generations. Based on the HEP algorithm, new routines for increasing and decreasing population size are designed, in order to increase the diversity and remove redundancy respectively. These routines work by comparing the dissimilarity of individuals in the population. We have also defined a structural dissimilarity comparison metric for comparing different Bayesian network structures. In this section, the techniques will be described in detail.

3.1 Structural Dissimilarity Comparison

The objective of the A-HEP algorithm is to speed up the process of searching good network structures with small MDL scores. Therefore, a representation for network structures is defined. In A-HEP, a network structure is represented as a two-dimensional matrix (shown as Fig. 2). The size of the matrix is $n \times n$, where n is the number of nodes.



	A	B	C	D	E
A	0	0	0	0	0
B	0	0	0	0	0
C	1	1	0	0	0
D	0	1	0	0	0
E	0	0	0	1	1

Fig. 2. Representation of a Bayesian network structure

The value of the matrix is define by:

$$StructureB_{ij} = \begin{cases} 1 & \text{if node } j \text{ is the parent of node } i \\ 0 & \text{if node } j \text{ is } \textit{not} \text{ the parent of node } i \end{cases}$$

Since the individuals are represented in this data structure, we can define a function to compare the distance between two individuals, i.e. the dissimilarity of two Bayesian networks.

$$Distance(B,B') = \sum_{i,j}^{n} x_{ij} \qquad \begin{cases} \text{where } x_{ij} = 0 \text{ if } B_{ij} = B'_{ij} \\ \text{where } x_{ij} = 1 \text{ if } B_{ij} \neq B'_{ij} \end{cases}$$

For example, the distance between the two Bayesian networks in Fig. 3 is 2, as one edge is added and one edge is deleted.

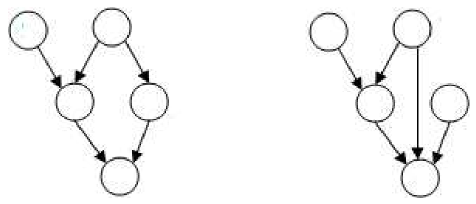


Fig. 3. Two different Bayesian network structures with *distance* = 2.

3.2 Dynamic Population Size

In the original HEP algorithm, the population size m is fixed. In each generation, each individual either merges with another individual in the previous generation or mutates itself by different operators to get its new offspring. New population size is then increased to $2m$. After that, a number of pairwise competitions are carried out and the fittest m individuals are kept for the next generation.

In A-HEP, the population size changes adaptively according to the dissimilarity of individuals in the current population and newly evolved offspring. The principles of this approach are:

1. If new offspring is quite different from the individuals of the current population (i.e. the distance is large), it is worthy to keep it, although it has worse fitness.
2. If some individuals are similar or exactly the same as other individuals (i.e. the distance is small or zero), it is reasonable to remove them.

Undoubtedly, the above principles cannot be applied strictly throughout the whole evolution process. Otherwise, search space for better solutions is limited by the second principle, and the algorithm does not converge under the first principle. Therefore, the stage of evolution must be considered as a factor of population size expansion and contraction. In the early stage of evolution, we can allow more degree of population size expansion and less degree of contraction. In the later stage of evolution, we can limit population size expansion and allow removing more redundant individuals in the population. At the same time, population size is maintained within a range of values.

A-HEP is developed based on the original HEP with several modifications. The original CI Test Phase and different operators are still used in the new algorithm. An increasing routine and a decreasing routine are introduced for processing mutated individuals to change the population size adaptively. In A-HEP, the pairwise competition is no longer used because new routines are employed to decide which individuals should be kept.

Algorithm 2 outlines the operations of A-HEP. The following notations are used to describe A-HEP throughout this paper:

p_c is the current population size in this generation.
p_{new} is the new population size for next generation.
p_{max} is the maximum population size.
p_{min} is the minimum population size.
Gen_c is the current generation number.
Gen_{total} is the total generation number.
$AvgDis_i$ is average distance between the individual I_i to all other individuals.
R_1, R_2, R_3 are three random numbers between 0 and 1.

Algorithm 2 Algorithm of A-HEP

```
CI Test Phase
Evolutionary Programming Search Space
  Set $Gen_c = 0$;
  Initialize and evaluate the population with size $p_{init}$;
  while $Gen_c < Gen_{total}$ do
    Randomly select $p_c/2$ individuals for merge operator
    Each unselected and unmerged individual produce one offspring by different mutation operators
    Increasing Routine (See Routine 1);
    Decreasing Routine (See Routine 2);
    Update population size, $p_c = p_{new}$
  end while
  Return the final structure with the lowest MDL score.
```

Increasing Routine: In order to increase the diversity at the early stage of evolution, the population size is increased adaptively by examining the new mutated offspring. If it is very different from the individuals in current population, the parent and itself are kept, no matter what their fitness values are. This technique can prevent premature convergence. However, the population size expansion must be controlled by considering the following factors:

1. Ratio of the current population size to the maximum population size
2. Ratio of the current generation number to the total generation number

In order to decide the fate of the new mutated offspring and its parent, calculation is done on the average distance between them and all the other individuals in the current population by the dissimilarity metric defined in previous section. If it is larger than a threshold, *far-factor* $\times$ *no. of nodes*, both of them are preserved for next generation. Since the size of matrix for representing a Bayesian network structure depends on its number of attributes, the distance threshold should also depend on the number of nodes.

Routine 1 For increasing population size

```
$p_{new} \leftarrow p_c$
$i \leftarrow 0$;
while $p_c < p_{max}$, and $R_1 > p_c/p_{max}$, and $R_2 > Gen_c/Gen_{total}$, and $i < p_c$ do
  for each mutated offspring $I_i$ do
    Calculate $AvgDis_i$;
    if $AvgDis_i >$ far-factor $\times$ no. of nodes then
      Both its parent and itself are kept for next generation;
      $p_{new} \leftarrow p_{new} + 1$;
    end if
  end for
  $i \leftarrow i + 1$;
end while
```

With the population size increasing routine, the diversity increases with the search space in the early generations. Experimental results have shown that better network structures can be obtained.

Decreasing Routine: The main objective of A-HEP is to reduce the running time of the original HEP. This can be achieved by removing the *redundant* individuals in the population at the later part of evolution. Based on AEGA [5], a routine is designed for decreasing the population size adaptively by considering the dissimilarity between individuals.

Similar to the increasing routine, ratio of the current generation number to the total generation number is used as a parameter to delay the time of population size contraction. There are two cases when population size is going to decrease:

1. When two mutated offsprings are fitter, more similar between themselves than their parents be, and their distance is closer than the threshold, *cutoff-distance*;
2. The pair of chosen individuals for next generation are exactly the same;

With the decreasing routine, redundant individuals are removed in the population at the later stage of evolution. As the population size decreases, the computation effort required for each generation is also reduced. Consequently, the time it takes to obtain the final structure is greatly reduced. This can be proved by experiments in next section.

Routine 2 For decreasing population size

```
for each pair of mutated individuals $I_i$,$I_j$ and their parents $I_i^p$,$I_j^p$ do
  Calculate $d_1 = Distance(I_i^p, I_j^p)$;
  Calculate $d_2 = Distance(I_i, I_j)$;
  Compare the fitness of $I_i$ with $I_i^p$ and $I_j$ with $I_j^p$, and take the fitter one in each
  pair for next generation;
  if Both children,$I_i, I_j$, are chosen then
    if $p_c > p_{min}$ and $d2 < d1$ and $d1 <$ cutoff-distance then
      Choose the fitter child and remove another one;
      $p_{new} \leftarrow p_{new} - 1$;
    end if
  end if
  if $Distance$ between chosen pair = 0, and $p_c > p_{min}$, and $R_3 > 1 - Gen_c/Gen_{total}$
  then
    Remove one of them;
    $p_{new} \leftarrow p_{new} - 1$;
  end if
end for
```

4 Evaluation on Proposed Algorithm

The performance of A-HEP was evaluated and compared to the original HEP by the following set of experiments. An optimized algorithm should obtain a Bayesian network with comparable or even better quality in shorter time. Therefore, the following experiments are used to examine the performance of A-HEP on running time and fitness of final network structure obtained.

4.1 Experimental Methodology

In our experiments, we used seven data sets generated from the well-known benchmarks of Bayesian networks including the ALARM, the PRINTD, and the ASIA networks. Alarm1000, alarm2000, alarm5000, alarm10000, and alarm-O were created from the ALARM network. These data sets were obtained from two different sources. One of them (alarm-O) containing 10,000 cases was obtained from Bayesian Network PowerConstructor [1]. The others were used for evaluating MDLEP [7]. The four data sets are of different sizes and contain 1,000, 2,000, 5,000, and 10,000 cases respectively.

The asia1000 data set with 1,000 cases is generated from the ASIA network and the last data set containing 5,000 cases is created from the PRINTD network. Since both algorithm are stochastic in nature, we conducted 40 trials for each experiment. The programs were executed on the same Nix dual Intel Xeon 2.2GHz Linux machine. For both algorithms, we set $\triangle_\alpha$ to be 0.02 and the initial population size to 50. For A-HEP, the maximum and the minimum population sizes were 100 and 3 respectively. The *far-factor* and *cutoff-distance* were set to be 0.8 and 1. The maximum number of generations is 5000. The results are summarized in Fig. 4 and Table 1.

Table 1. Performance comparison between HEP nd A-HEP

Data set	Average Running Time(second)			Average MDL Score of final network		
	HEP	A-HEP	Ratio (A-HEP/HEP)	HEP	A-HEP	Difference (A-HEP - HEP)
alarm1000	53.08 ±1.14	20.10 ±4.67	0.38	17862.48 ±19.68	17877.48 ±28.31	15.01 (0.08%)
alarm2000	56.05 ±0.51	23.70 ±6.01	0.42	33773.05 ±3.14	33787.45 ±56.24	14.39 (0.04%)
alarm5000	63.38 ±0.9	47.68 ±11.07	0.75	81004.00 ±0.0	81015.22 ±68.18	11.22 (0.01%)
alarm10000	75.97 ±1.84	54.50 ±5.92	0.72	158517.54 ±247.02	158473.53 ±90.45	-44.01 (-0.28%)
alarm-O	102.65 ±25.25	54.33 ±11.18	0.53	138549.48 ±385.83	138564.95 ±405.60	15.48 (≈0%)
asia1000	8.18 ±0.38	3.40 ±0.50	0.42	3398.66 ±0.16	3398.60 ±0.00	-0.06 (≈0%)
printd5000	38.33 ±0.62	14.30 ±1.34	0.37	106542.00 ±0.00	106542.00 ±0.00	0.00 (0%)
Average	-	-	0.51	-	-	1.72

4.2 Comparison on Running Time

In our experiments, the running time of A-HEP is 20-60% less than the original HEP. This improvement is due to the adaptive population size concept. Once the population converges to a certain degree of similarity, redundant individuals are removed from the population. Therefore, the computation effort for later generations are greatly reduced and the running time is shortened. Table 1 shows the average running time for each data set. Average improvement of A-HEP is around 1.89 times faster.

This speed-up is particular significant for small data sets, such as asia1000 and alarm1000. At the later stage of evolution, the individuals in the population are very similar or even exactly the same. The decreasing routine removes these individuals from the population dynamically, and increases the efficiency of the algorithm. By adjusting the far-factor and population size limits, the running time can be further reduced.

4.3 Comparison on Fitness of Final Network

Although A-HEP can learn the Bayesian network structure in a shorter time, the quality of the final networks are also important. Therefore, the average fitness (MDL score) of the best network obtained by each algorithm are compared. Table 1 shows the MDL score of final network obtained by both algorithms. In real trials, both algorithms can obtain the individual with minimum MDL score in most cases. The differences of MDL score on Table 1 is mainly due to obtaining sub-optimal structures in particular trials. However, those MDL score

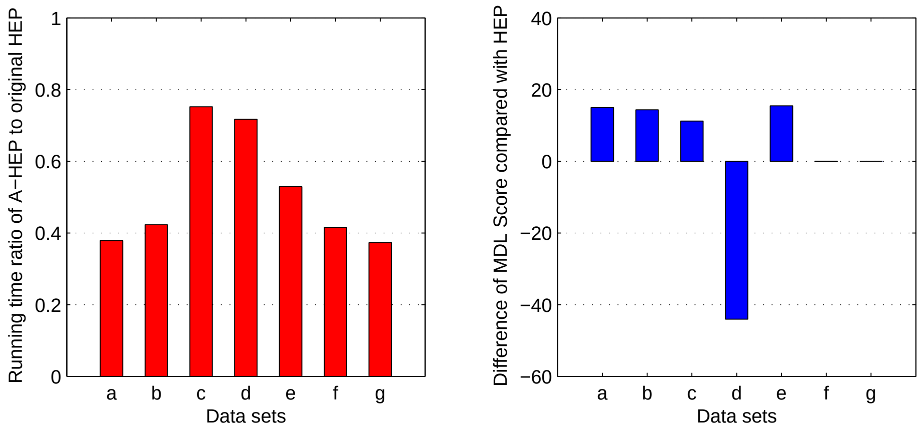


Fig. 4. Data set (a)alarm-1000, (b)alarm-2000, (c)alarm-5000, (d)alarm-10000, (e)alarm-O (original), (f)asia-1000 and (g)printd-5000; Left: Comparison on average running time in each data set; Right: Comparison on the average fitness (MDL score) of final Bayesian network obtained

differences are insignificant. A-HEP can even obtain fitter Bayesian network in more cases than HEP in alarm-10000 data set. We can conclude that A-HEP has comparable BN structure learning performance as HEP. With shorter running time and comparable quality of final network structure obtained, A-HEP is more efficient than the state-of-art Bayesian network learning algorithm - HEP.

5 Conclusion and Future Work

In this paper, we have presented the adaptive population size evolutionary algorithm, A-HEP, for learning Bayesian network structures. This is an optimized version of HEP which is one of the state-of-art algorithms of this type. The technique is based on the concept of adjusting population size adaptively according to the dissimilarity of individuals in current population. With the use of increasing and decreasing routines, the population expands in early generations for increasing diversity, and contracts in later generations for reducing computation time. A-HEP has been experimentally tested with several data sets of different sizes and numbers of variables. The performance of it is compared with the original HEP on running time and quality of Bayesian network obtained. All experiments have demonstrated that A-HEP consistently and significantly reduces the running time with comparable performance on Bayesian network learning. This speed-up is very important as A-HEP can be used efficiently for learning Bayesian networks on many data mining problems.

In order to further optimize the learning performance of A-HEP, we are going to design a new operator for crossover between two individuals. We also want to investigate the feasibility of applying dynamic population size concept in other algorithms.

References

1. Jie Cheng, Russel Greiner, J. Kelly, D. Bell, and W. Liu. Learning Bayesian network from data: An information-theory based approached. *Artificial Intelligence*, 137:43–90, 2002.
2. D. Heckerman. A tutorial on learning Bayesian networks. Technical report, Microsoft Research, Advanced Technology Division, March 1995.
3. W. Lam and F. Bacchus. Learning Bayesian belief networks-an approach based on the MDL principle. *Computational Intelligence*, 10(4):269–293, 1994.
4. 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.
5. Kwong Sak Leung and Y. Liang. Adaptive elitist-population based genetic algorithm. In *Genetic and Evolutionary Computation Conference*, volume LNCS 2723, pages 1160–1171, 2003.
6. Judea Pearl. *Probabilistic Reasoning in Intelligent Systems : Networks of Plausible Inference*. Morgan Kaufmann, 1988.
7. Man Leung Wong, Wai Lam, and Kwong Sak 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.
8. Man Leung Wong, Shing Yan Lee, and Kwong Sak Leung. A hybrid approach to discover Bayesian networks from databases using evolutionary programming. In *Proceedings of the 2002 IEEE International Conference on Data Mining*, pages 498–505, 2002.