Soft Comput (2017) 21:1065–1080
DOI 10.1007/s00500-015-1841-z

METHODOLOGIES AND APPLICATION

Ensemble bayesian networks evolved with speciation for high-performance prediction in data mining

Kyung-Joong Kim[1] · **Sung-Bae Cho**[2]

Published online: 20 August 2015

Abstract Bayesian networks (BNs) can be easily refined (or learn) using data given prior knowledge about a changing environment. Furthermore, by exploring multiple diverse BNs in parallel, it is expected that an intelligent system may adapt quickly to changes in the environment, resulting in robust prediction. Recently, there have been attempts to design BN structures using evolutionary algorithms; however, most of these have used only the fittest solution from the final generation. Because it is difficult to combine all of the important factors into a single evaluation function, the solution is often biased and of limited adaptability. Here we describe a method of generating diverse BN structures via speciation and selective combination for adaptive prediction. Experiments using the seven benchmark networks show that the proposed method can result in improved accuracy in handling uncertainty by exploiting ensembles of BNs evolved by speciation.

Keywords Prediction · Bayesian networks · Uncertainty · Ensemble · Speciation · Evolution

Communicated by V. Loia.

✉ Kyung-Joong Kim
kimkj@sejong.ac.kr

Sung-Bae Cho
sbcho@cs.yonsei.ac.kr

[1] Department of Computer Science and Engineering, Sejong University, Seoul, Korea

[2] Department of Computer Science, Yonsei University, Seoul, Korea

1 Introduction

Bayesian networks (BNs) are a commonly used approach to deal with uncertainty, and represent joint probability distributions of a domain. BNs and the associated schemes constitute a probabilistic framework for reasoning with uncertainty, and in recent years have gained popularity in the field of artificial intelligence Barber (2012). A BN is a directed acyclic graph (DAG), where the nodes are random variables and the lack of arcs specifies conditional independence between variables. BNs are typically constructed manually by experts or learned from data (Shen 2009; Koller and Friedman 2009).

It is not straightforward to determine the BN that best reflects a particular problem from a database of cases because of the large number of possible DAG structures, given even a small number of nodes to connect (Daly et al. 2011; Gamez et al. 2011). Consequently, there have been a number of reports of heuristic search techniques to identify good models (Gamez et al. 2011; Cooper and Herskovits 1992). Recently, methods to design BN structures using evolutionary algorithms have appeared (Larranaga et al. 2013); however, these have mostly used only the fittest solution in the final generation (Larranaga et al. 1996a, b; Wong et al. 1999).

The rationale that combining multiple diverse models can perform better than a single model is based on evidence from ensemble research (Luo et al. 2011; Peng et al. 2011). Evolutionary computation is suitable for generating multiple models because it is a population-based search method (Kim and Cho 2012). Standard genetic algorithms, however, tend to generate solutions that are not very diverse because of genetic drift, and there is little benefit from the combination of the similar models (Kim and Mckay 2012). Evolving multiple diverse solutions using speciation techniques can enhance the diversity of a population, and form better ensembles than using standard genetic algorithms (Kim et al. 2011). There

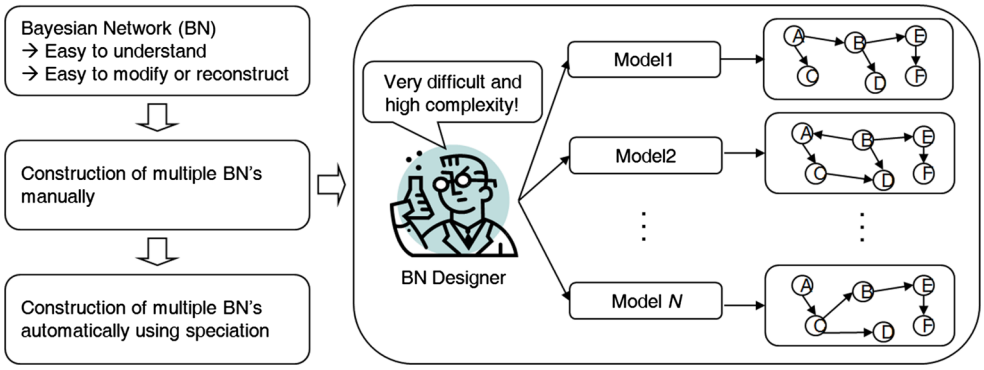

Fig. 1 The motivation behind this work. BNs are useful tools to represent uncertain knowledge; however, they can be difficult to construct. The situation becomes worse if multiple BNs are required. This motivates the BN designer to use an automated algorithm to devise multiple Bayesian networks using data

have been some reports of evolving multiple neural networks using speciation techniques and combining them to achieve improved performance (Kim and Cho 2005, 2008). These works focused on ensembles of diverse evolutionary neural networks for classification problems.

In this work, we evolve multiple evolutionary BNs to form a favorable predictive model, as shown in Fig. 1. Some of the evolved BNs may be expected to exhibit abnormal behavior; however, other BNs may compensate for these shortcomings by providing a correct prediction. In our learning procedure, densely populated areas are penalized for overpopulation, and candidates in less populated spaces generate more offspring in the subsequent generation. In this manner, the overall population diversity is maintained during the evolution. Following learning, ensemble members are selected from the pool of BNs in the final generation. An ensemble of the members generates a consensus for several target nodes for new input data.

There have been relatively few reports of the construction of ensembles of BNs. Here we investigate ensembles of Bayesian networks with several different evolutionary algorithms and heuristics. The use of "expert-style" ensembles (i.e., modular BNs) is shown to be effective in BN ensembles (Hwang and Cho 2009). Here, the aim was to develop a framework to build an ensemble of BNs. Because the model is somewhat different from other classification algorithms, it is beneficial to propose guidelines to build the ensembles using efficient techniques. We combine evolutionary computation with simple heuristic ensemble search techniques. The results show that the framework is effective in predicting diagnostic nodes.

The method was evaluated in terms of the prediction accuracy rather than structural similarity, which has been widely used (Larranaga et al. 1996a). This is because it is not easy to define the structural similarity between a single network and an ensemble of networks. The proposed method was compared with standard genetic algorithms and the greedy-style K2 algorithm (Cooper and Herskovits 1992).

The remainder of the paper is organized as follows. Section 2 describes the background, including research into evolutionary BNs. Section 3 describes evolutionary learning with speciation, and the formation of selective ensembles. Section 4 describes the experimental results and provides some analysis.

2 Background

2.1 Bayesian networks

BNs can be used for inference and representation of an environment in the presence of uncertain information (Barber 2012). The nodes of a BN represent random variables, and the lack of arcs represents the conditional independence between variables. In addition to the network structure, the conditional probability distribution of the nodes must be specified. The structure is typically either designed by experts or learned from data. Given observed data, the probability of the state of unknown diagnostic nodes can be computed using an inference algorithm.

We use $\langle B, \theta_B \rangle$ to denote a BN with a structure B and associated conditional probabilities θ_B. $P\langle B, \theta_B \rangle$ denotes the joint probability distribution of all the variables of this network. A BN is a DAG $B = (V, E)$, where the set of nodes $V = \{x_1, x_2, ..., x_n\}$ represents the domain variables and E provides information on conditional independence. For each variable $x_i \in V$, the conditional probability distribution is $P(x_i|\mathrm{Pa}(x_i))$, where $\mathrm{Pa}(x_i)$ represents the parent set of variable x_i, i.e.,

$$P\langle B, \theta_B \rangle = P(x_1, x_2, ..., x_n) = \prod_{i=1}^{n} P(x_i|\mathrm{Pa}(x_i)) \quad (1)$$

2.2 Learning Bayesian structures using evolutionary computation

Evolutionary computation is useful for global searches because it maintains a large population of candidates by exploring a search space using genetic operators (i.e., mutation and crossover) (Goldberg 2008). However, it also suffers from premature convergence or genetic drift due to the dominance of super individuals (i.e., the fittest solution of an early generation) and requires additional techniques to maintain the diversity of a population (Rogers and Prugel-Bennett 1999).

One important aspect of learning with BNs using data is the scoring metric to evaluate the goodness of a given candidate network for the database, and applying a search procedure to explore the set of candidate networks. Because learning with BNs is, in general, an NP-hard problem (Chickering et al. 1994), exact methods are not feasible (Heckerman 2008). Recently, however, there have been a number of reports of evolutionary computation as a search heuristic for this problem, as listed in Table 1.

Wong et al. (2004) used a cooperative co-evolutionary genetic algorithm (GA) to create BNs with a learned structure. They transformed the learning of BNs into a number of sub-problems, and combined the solutions from subpopulations. Although they considered speciation in the learning, the final outcome was not an ensemble of BNs. Li et al. (2005) applied speciation based on crowding to evolutionary programming for learned BN structures; however, the main purpose of this research was to avoid premature convergence, rather than search ensembles. Kim et al. (2005) applied fitness sharing to the ASIA network benchmarking problem, and Muruzabal and Cotta (2007) examined a number of evolutionary programming algorithms for BN induction problems.

2.3 Ensemble Bayesian networks

There have been a number of reports of combining multiple BNs that have focused on learning for classification, including speaker identification and protein secondary structure prediction (Robles et al. 2004), as well as regression, including estimation of user preferences (Feng et al. 2014), and modeling complex interactions, including regulatory pathways, and context awareness (Hwang and Cho 2009).

Su and Khoshgoftaar (2008) applied multiple Bayesian networks to collaborative filtering (CF) tasks in real-world multi-class CF data. Garg et al. (2003) developed a supervised learning framework for BNs, which was based on the AdaBoost algorithm of Schapire and Freund. Their framework covered static and dynamic BNs, with both discrete and continuous states. They tested the framework with a novel multimodal human–computer interaction (HCI) application, i.e., a speech-based command and control interface for a smart kiosk.

Table 1 Summary of research into evolutionary BNs

References	Representation	Type of EA	Score metric	Ensemble	Dataset
Myers et al. (1999)	Connection matrix	GA	K2 Metric	–	ASIA
Li et al. (2005)	DAG	EP	MDL	–	ALARM
Wong et al. (2004)	Connection matrix	CCGA	MDL	–	ALARM, PRINTD
Wong et al. (1999)	DAG	EP	MDL	–	ALARM, PRINTD
Larranaga et al. (1996b)	Variable order list	GA	K2 Metric	–	ALARM
Larranaga et al. (1996a)	Connection matrix	GA	K2 metric	–	ASIA, ALARM
Kim et al. (2005)	Connection matrix + variable order	GA, FSGA	DPSM	Selective ensemble (clustering)	ASIA
The proposed method	Connection matrix	GA, FSGA, DCGA	K2 metric	Selective ensemble ($_{50}C_3$, Greedy, and Expert)	Cancer, earthquake, survey, ASIA, insurance, water, ALARM

In the evolutionary algorithm, each BN is encoded as a matrix, list, or graph. There are several types of evolutionary algorithm, including GAs and EP. The choice of algorithm affects the implementation of the evolutionary BN

DPSM Dirichlet prior score metric, *MDL* minimum description length, *K2 Metric* Bayesian–Dirichlet score with uniform priors, *EA* evolutionary algorithm, *CCGA* cooperative co-evolutionary genetic algorithm, *EP* evolutionary programming, *FSGA* fitness sharing genetic algorithm, *DCGA* deterministic crowding genetic algorithm

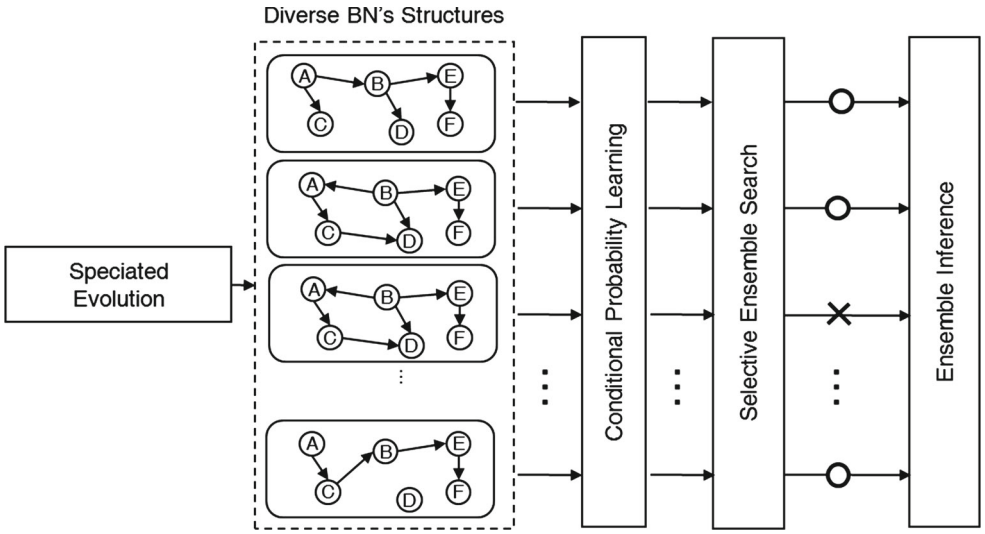

Fig. 2 An overview of the proposed method. Speciation helps maintain the diversity of the population in the evolutionary optimization. The next step is to train the conditional probability table (CPT) of the networks. A selection of the trained networks are combined to produce the final inference

Robles et al. (2004) used three different BN structures: naïve Bayes, interval estimation naïve Bayes (IENB) and Pazzani's model of joining attributes in naïve Bayes to build level-1 classifiers in a stacked generalization scheme. With that scheme, a number of classifier layers were designed from part of a global multi-classifier, where the upper layer classifiers receive class predicted from the previous layer as the input. The predictive accuracy was found to outperform the best secondary structure predictors by 1.21 % on average. Li et al. (2008) proposed a method of BN combination without loss of any information, and freedom of datasets from the graphical characterization of global models. Their approach was a kind of structural combination of multiple local models, resulting in a global model.

Pena et al. (2004) trained a diverse set of BN models, and isolated recurring features from multiple locally optimal models. In the work, their goal was to assist the user in interpreting the Bayesian network models. They ran k-greedy equivalence search repeatedly, extracted features from them, and evaluated confidence of features for users. Although their work was based on multiple Bayesian networks, they focused on the interpretation of models for user instead of prediction.

Feng et al. (2014) proposed a method to combine multiple BNs and carried out a theoretical analysis showing the distinctive advantage of this combination. In addition, the combination approach was applied to recommendation systems, bank direct marketing, and disease diagnosis. Hu and Wang (2013) found that there were a large number of BN learning algorithms; however, the accuracy of these was poor because of a lack of sufficient microarray data. They proposed to combine BNs from relevant literature and learning from microarray data. Hwang and Cho (2009) proposed a design for a very complex BN from modularized BNs using life-log data. They constructed a BN with 588 arcs and 462 nodes using 39 designed BNs.

3 Method

The proposed algorithm is composed of two stages: evolutionary learning with speciation, and the selective combination of the resulting BNs. Figure 2 shows an illustration of the algorithm. The purpose of the first stage is to evolve multiple BNs, which should ideally have differing characteristics. In this work, speciation algorithms that promote the diversity of a population were applied to avoid genetic drift (i.e., premature convergence of the solutions, which may occur with a conventional GA).

Because the evolutionary algorithm produces a population of solutions following learning, it provides us with a pool of candidate BNs. The second stage combines a subset of the evolved BNs. Instead of using all of the BNs, some are selectively chosen for the final combination, because a combination of subsets provides favorable performance compared with an ensemble of all members (Zhou 2012). The goodness (prediction accuracy) of candidate ensembles is measured by the performance on the training datasets in the ensemble search procedure, and the optimal ensemble is determined based on test datasets.

3.1 Overview

The structure of a BN is encoded into a matrix and optimized using GAs. It is a population-based search that maintains multiple solutions (i.e., multiple structures of the BNs). Because it provides a population of candidate BNs, it is

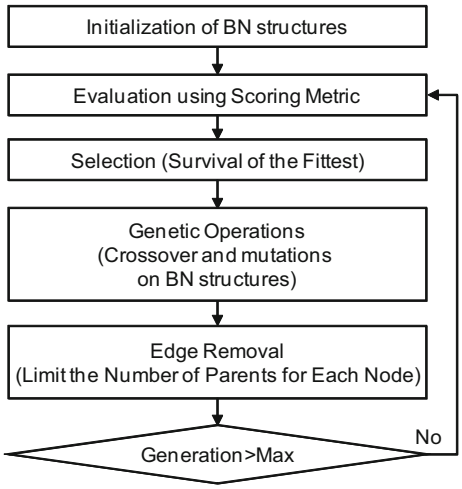

Fig. 3 The BN search process using a GA. The purpose of the algorithm is to optimize the topology of the BN for the training data. Selection, crossover, and mutation operators are used, and the edge removal operator is used to limit the number of parents for each node

promising to combine subsets to achieve accurate prediction. If all the BNs in the population were identical, it is not expected that any synergism would result; however, because GAs suffer from the phenomenon of genetic drift (Kim and Mckay 2012), the population may be dominated by a single premature solution in the early stages of its evolution. Speciation helps maintain the diversity of a population by penalizing highly populated regions of the evolutionary space by discarding similar candidates from the population (Kim et al. 2011).

Figure 3 shows a flowchart describing the evolution of the BNs. An initial population of BN structures is randomly created, and the fitness (goodness) of the structure is evaluated using scoring metrics based on the training data. During the selection stage, each network has a different selection pressure (chance of selection), which is proportional to the fitness. Several genetic operators are applied to generate a new population of candidates by randomly exchanging a proportion of structures or changing some edges. The number of parents for each node is limited to a pre-defined maximum to avoid excessively long evaluation times. An edge removal operator was used to delete edges with a local optimizer (Larranaga et al. 1996a). The final BNs become a new population for the subsequent generation. Evaluation, selection, genetic operation, and repair are repeated until the maximum number of generations is reached.

3.2 Representation

We used a connection matrix representation, which maintains the connectivity information between two nodes (Larranaga et al. 1996a). In this representation, a BN structure is represented by an $n \times n$ connectivity matrix C. Elements of

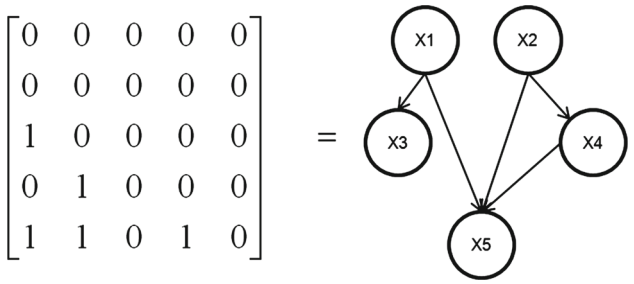

Fig. 4 An example connection matrix representation of a BN. Each matrix encodes a BN topology, and the values are optimized using the evolutionary algorithm. Because the value of an element c_{31} is '1,' there is an arc from x_1 to x_3. The number of edges in the network is equal to the number of elements in the matrix that are equal to one

the matrix C that are equal to one are used to represent the existence of arcs between variables, i.e.,

$$c_{ij} = \begin{cases} 1 \text{ if there is an arc between } x_i \text{ and } x_j \ (i > j), \\ 0 \text{ otherwise.} \end{cases} \quad (2)$$

Figure 4 shows an example of decoding a BN using a connectivity matrix. The connection matrix represents the existence of arc among variables. Our approach is to construct the network using training data, and is based on data-driven induction of the BNs. The algorithm optimizes a subset of arcs that produce the maximum "Bayesian score metric" given the training data. It starts by randomly creating BNs, but gradually improves the topology using the GAs.

3.3 Evaluation

During evaluation, the fitness of each BN is evaluated using the Bayesian–Dirichlet score with uniform priors (K2 metric) with training data (Cooper and Herskovits 1992) as follows.

$$\log(P(B, D)) = \sum_{i=1}^{n} \sum_{j=1}^{q_i} \left(\log\left(\frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!}\right) + \sum_{k=1}^{r_i} \log(N_{ijk}!) \right) N_{ij} = \sum_{k=1}^{r_i} N_{ijk}. \quad (3)$$

where r_i is the number of possible values of variable x_i, q_i is the number of possible configurations (instantiations) for the variables in $\mathrm{Pa}(x_i)$, and N_{ijk} is the number of cases in the training data in which variable x_i has its kth value and $\mathrm{Pa}(x_i)$ is instantiated to the jth value. The factorials required to compute the K2 metric have been pre-computed and have been stored in an array (from 0 to $(m+r-1)!$) where m is the number of cases in the training data. The scoring metric is in the range $-\infty$ to $+\infty$, and a large positive value corresponds to good performance. However, it is usually negative value because of the logarithmic summation.

3.4 Genetic operations

Genetic operations include crossover and mutation (Goldberg 2008). The $n \times n$ matrix can be transformed into a one-dimensional (1D) array of size n^2. For example, the matrix in Figure 4 is converted into 00000 00000 10000 01000 11010 by concatenating the horizontal vectors of the matrix, and each segment with n bits encodes the parents of the node so that n consecutive bits encode the data of the child nodes for each node. (We use horizontal vector concatenation, although an alternative is to transform the matrix into a 1D array by concatenating the vertical vectors of the matrix.)

We use a simple one-point crossover for the 1D array representation. From two parents chosen at random, parts of the genetic information are exchanged using the crossover operation. Crossover and mutation are applied sequentially. There are two types of mutation: arc addition and arc deletion. During mutation, an entry for the matrix is chosen at random, and a different type of mutation is applied depending on the existence of an edge. If there is a link between two nodes, the arc is deleted, whereas if there is no link, an arc is created. The probability of mutation is relatively low. Based on the prior knowledge of the variable order, the crossover and mutation were designed not to produce the cases that are not consistent with the topological order of variables.

The number of parents for each node is a crucial factor affecting the evaluation time, and the large number of parents can form a bottleneck in the evolution. The number of parents for each node was limited to a predefined maximum. If the genetic operations result in an individual with a number of parents that is larger than the upper bound u, we must eliminate (an edge removal operator) a subset of parents. Because the performance of random subset selection is poor, we adopted a local optimizer to choose the best subsets to retain among the candidates. We choose the best subset of parents (where the size of the subset is smaller than the maximum) based on the K2 metric scores.

3.5 Speciation

We used two representative speciation methods: fitness sharing and deterministic crowding (Goldberg 2008; Mahfoud 1995). During speciation, it is necessary to measure the similarity between two BNs. Initially, the representation (matrix) of each BN is converted into a 1D array. The Hamming distance between the two arrays is calculated to describe the similarity (Note that this considers only the structural similarity).

In the fitness sharing approach, the original fitness of the candidates is adjusted by considering a penalty for overpopulation. The penalty is proportionate to the number of individuals and the closeness (this is termed the sharing radius σ_s). The population of BNs is defined as

$$\{B_1, B_2, \ldots, B_{\text{pop\_size}}\} \tag{4}$$

Given that f_i is the fitness of an individual B_i and $\text{sh}(d_{ij})$ is a sharing function, the shared fitness fs_i is computed as follows:

$$\text{fs}_i = \frac{f_i}{\sum_{j=1}^{\text{pop\_size}} \text{sh}(d_{ij})} \tag{5}$$

The sharing function $\text{sh}(d_{ij})$ is computed using the distance d_{ij}, which corresponds to the difference between individuals B_i and B_j, and is defined as follows. The σ_s was calculate based on the equation from Kim and Cho (2008).

$$\begin{aligned} \text{sh}(d_{ij}) &= \begin{cases} 1 - \frac{d_{ij}}{\sigma_s}, & 0 \le d_{ij} < \sigma_s \\ 0, & d_{ij} \ge \sigma_s \end{cases} \\ d_{ij} &= \text{Hamming}(B_i, B_j) \\ \sigma_s &= \frac{1}{2 \times \text{pop\_size} \times (\text{pop\_size}-1)} \sum_{i=1}^{\text{pop\_size}} \sum_{j=1, i \neq j}^{\text{pop\_size}} d_{ij} \end{aligned} \tag{6}$$

The deterministic crowding GA was proposed by Mahfoud (1995) and has been widely used in various domains Kim and Cho (2005). The principle is to use competition between two similar individuals, whereby only one survives into the next generation based on the results of the league. During the shuffling step, the population index of individuals is randomized so that the first two individuals B_i and B_j generate two offspring using the genetic operations. We now have four individuals: the two parents B_i and B_j, and two offspring $B_{i+\text{pop\_size}}$ and $B_{j+\text{pop\_size}}$. We calculate the Hamming distance between BNs; if $d(B_i,B_{i+\text{pop\_size}}) + d(B_j,B_{j+\text{pop\_size}}) < d(B_i,B_{j+\text{pop\_size}}) + d(B_j,B_{i+\text{pop\_size}})$, we have two competitions (B_i vs. B_{i+pop_size} and B_j vs. B_{j+pop_size}), and vice versa. For each competition, only one individual survives to the next generation based on the fitness. This causes similar individuals to compete with each other, and increases the diversity of the population. This procedure is repeated for the remaining individuals.

3.6 Ensemble search and combination

During this stage, conditional probability tables of the BNs in the final generation are learned from the training data and the best ensemble is optimized using the following three heuristics: ${}_{50}C_3$ search, greedy search and expert. The ${}_{50}C_3$ search locates the best ensemble exhaustively by enumerating all possible candidates. Because the search space is potentially huge, we restrict the size of the ensemble (i.e., the number of members) to three. In total, there are $(\text{pop\_size})^3$ ensemble

```
E={}; // Ensemble
Temp={}, Best={}; // Temporary Space
Minimum=∞;    // Minimum Error
while (1) {
    for (i=0; i< POP_SIZE; i++) {
      Temp=E+{B_i};
      if (Minimum > Error(Temp)) { Minimum=Error(Temp); Best=Temp; }
    }
    if (E==BEST) break; else E=BEST;
}
```

Fig. 5 A pseudo-code representation of the greedy ensemble search. Here B_i is the ith BN. The process starts with an empty ensemble and gradually adds BNs to maximize the performance gain. The process continues until there is no further increase in performance

candidates. With the greedy search, the ensemble size is not fixed. It grows continuously if the new members are added to minimize the errors, as shown in Fig. 5.

The prediction error on the test data was defined as follows. Our aim is not classification but, rather, regression to produce similar outcomes to the original BN. We adopted the measure from the regression studies in the field of machine learning., i.e.,

$$\text{Error} = \sum_{i=1}^{\text{Samples}} \times \left(\text{BN}_{\text{estimated}}(\text{Target}|\text{E})_{\text{i}} - \text{BN}_{\text{original}}(\text{Target}|\text{E}_{\text{i}})\right)^2 \tag{7}$$

where E is evidence. BN represents the probability that is given by the network.

The expert method assigns one BN for each diagnostic node based on the performance on training cases. For example, if there are NBNs available and M diagnostic nodes, it calculates the prediction error of each BN for all diagnostic nodes using the training cases. Among the N BNs, only one BN is assigned to a diagnostic node for the prediction of unseen cases. The assigned BN is regarded as an "expert" on the node. As a result, the size of ensemble is dependent on the assignment of networks because one BN can be assigned to more than one diagnostic node. The total number of possible ensembles is N^M. In sum, in this algorithm, we select the best BN for each node based on errors from the training data, and divide the prediction tasks into a number of small tasks based on the target diagnostic node, and then assign a different expert for each BN. Each expert BN is used to predict the node of test cases.

There are a number of different ways to combine multiple BNs. For example, it is possible to construct a single BN from the multiple BNs (topological fusion); however, it is possible that conflict between the members of the ensemble may occur when attempting to generate a single complex model. Furthermore, it is possible that the final model will contain a large number of nodes and edges, which results in significant computational expense. In this study, we combine BNs by averaging the probabilities of each. This approach allows each network to run in parallel, and requires relatively little effort in the combination.

4 Results

The benchmark networks were downloaded from several BN repositories. In this work, we used some real-world networks (Insurance, Water, and Alarm) created by domain experts. We generated samples using benchmark networks with GENIE (GENIE 2015) and manually generated a node ordering using these benchmark structures (Cooper and Herskovits 1992; Larranaga et al. 1996a). A conditional probability table (CPT) was created, and learning and probabilistic inference was implemented using the SMILE C++ library (GENIE 2015). The proposed method was compared with the BN learning K2 algorithm (Larranaga et al. 1996a; Larranaga et al. 1996b). We focus on learning both the structure and CPTs of BNs, and so PC and EM algorithms are not suitable for the comparison. The diversity of the population was measured using the average Hamming distance between individuals. DCGA terminates when the diversity is less than one. Table 2 lists a summary of the details of this experiment. The upper bound of the number of states was chosen based on the model in Ref. Larranaga et al. (1996a). The parameters of genetic algorithm is as follows. The maximum number of generation, population size, crossover rate, and mutation rate are 1000, 50, 0.9, and 0.01, respectively.

Table 3 lists a summary of the results of the learning stage. The scoring metrics listed in the table are as follows. The average of fitness is given by

$$\text{Average} = \frac{1}{\text{pop\_size}} \sum_{i=1}^{\text{pop\_size}} f_i \tag{8}$$

Table 2 A description of the parameters for the networks is shown in Fig. 6

	Cancer	Earthquake	Survey	Asia	Insurance	Water	Alarm
References	Korb and Nicholson (2010)	Korb and Nicholson (2010)	Scutari and Denis (2014)	Lauritzen and Spiegelhalter (1988)	Binder et al. (1997)	Jensen et al. (1989)	Beinlich et al. (1989)
Variables	5	5	6	8	27	32	37
Number of target nodes	All nodes						8
Upper bound u of the number of parents	2				4		
Number of training cases	6000						
Number of test cases	6000						
Number of runs	5			10		5	10

With ALARM, only the diagnostic nodes in the original work were defined as the "target" node. The upper bound u is the maximum number of parents for each node. The results were averaged based on five runs. The maximum number of generations, population size, crossover rate, and mutation rate are for the evolutionary algorithm

Table 3 Comparison of the learning algorithms

Benchmark	K2 algorithm	GA	FSGA	DCGA
(a) Average of Bayesian scoring metric				
Cancer	−12,693	−12,696	−12,766	−12,693
Earthquake	−2602	−2614	−2617	−2602
Survey	−23,618	−23,621	−23,639	−23,618
Asia	−13,549	−13,627	−13,709	−13,550
Insurance	−80,660	−85,068	−84,699	−80,678
Water	−78,050	−79,756	−79,980	−78,052
Alarm	−57,635	−61,708	−61,841	−57,698
Average	−38,401	−39,870	−39,893	−38,413
(b) Diversity				
Cancer	0	0.8	0	0
Earthquake	0	2.2	4.6	0
Survey	0	0.9	7.9	0
Asia	0	5.8	10.6	0
Insurance	0	89.6	90.7	7.9
Water	0	120.5	132.3	85.7
Alarm	0	135.9	143.4	50.4
Average	0	50.8	55.6	20.6

K2 algorithm is a greedy-style learning algorithm, which compares the Bayesian scoring metric and the diversity of networks in the algorithms

and the diversity by

$$\text{Diversity} = \frac{2}{\text{pop\_size} \times (\text{pop\_size} - 1)} \times \sum_{i=1}^{\text{pop\_size}} \sum_{j=i+1}^{\text{pop\_size}} d(B_i, B_j) \quad (9)$$

The diversity measures normalized distance between all individuals (Gouvea and Araujo 2010). We find that DCGA and K2 algorithm gave the highest score in terms of fitness on the training cases. GA and FSGA scored lower in fitness; however, the diversity of population was high. The diversity of DCGA was zero for the small-sized network but relatively high in large networks.

Table 4 lists the errors in the ASIA network. With this network, all ensemble learning algorithms had small errors. However, the DCGA has no benefit from the ensemble because diversity of the method is zero in the network. The FSGA + Expert search was found to perform particularly well. The FSGA and DCGA speciation ensemble outperformed the K2 algorithm and GA ensemble. Statistical t-tests show that the FSGA + Expert algorithm (10.21 ± 0.45) outperform the K2 algorithm (17.46 ± 0.0), as well as the DCGA (12.10 ± 0.0) algorithms with a confidence interval of 99 %.

Table 5 lists the results of ALARM with test cases. ALARM is a large network and so it is difficult to learn using training data; the results show a relatively high error rate. With this network, the DCGA + Expert combination exhibited the best results. Unlike the smaller networks, the ensemble exhibited improved performance compared with that of the single-network approach. This is because it maintains a highly diverse population due to the complexity of the search space. Statistical t-tests show that the DCGA + Expert (24.60 ± 0.87) outperformed K2 algorithm (29.29 ± 0.0), GA (198.38 ± 126.43), FSGA (172.60 ± 52.47), and DCGA (28.88 ± 2.60) with a confidence interval of 99 %.

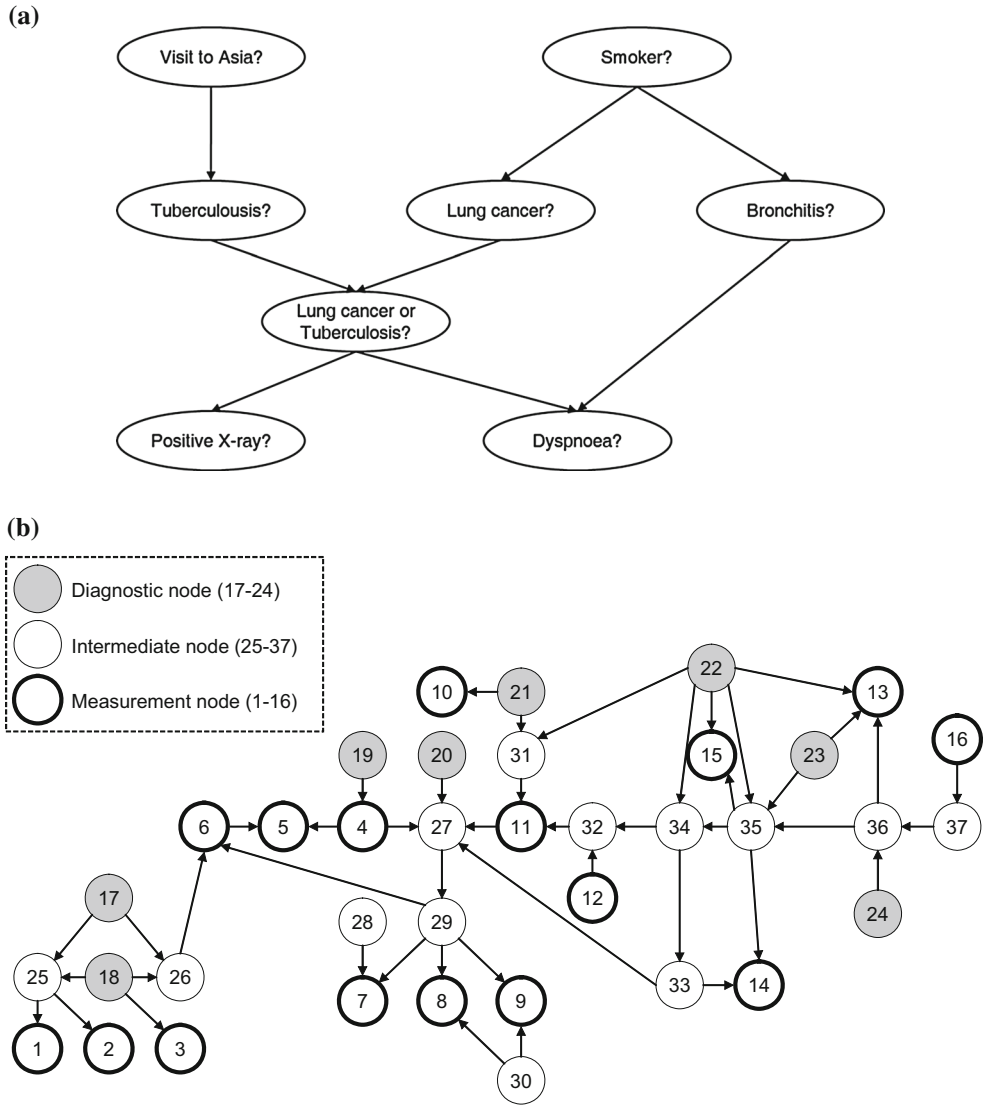

Fig. 6 The structures of networks used for benchmarking. **a** The ASIA network is a widely used small-scale BN benchmark. **b** The categorization of nodes in the ALARM network is from Ref. Beinlich et al. (1989). Sensory data from the intensive care unit (ICU) are used as inputs to the measurement node, and the network calculates the probability of the diagnostic states

The ALARM network combines seven BNs, as shown in Fig. 7. The individual networks were found to be the best predictor for some nodes (For example, the BN6 is the best one for "Anaphylaxis"). It combines some good BNs with a small number of inaccurate models. Although some of these BNs (for example, BN4, BN5, and BN6) had slightly higher error rates than others, it is interesting to note that they were essential in the ensemble system. For example, although the BN6 was not good for some nodes, it was highly competitive in others.

Table 6 shows the error rates of the single and ensemble BNs. It shows that the ensemble outperforms the single BN on all benchmark networks. In small networks (Cancer, Earthquake, Survey and Asia), the GA and FSGA ensembles show the lowest error rates. On the other hand, the DCGA ensembles always beat other algorithms in large networks (Insurance, Water, and Alarm). Among the three combination methods (Expert, Greedy and ${}_{50}C_3$), the Expert approach shows the lowest errors on most of networks except the Insurance (Greedy is the best). In sum, it is desirable to use the SGA + Expert or FSGA + Expert on small networks (5–8 nodes) and DCGA + Expert in large networks (27–37 nodes).

Table 7 shows the results of cross-validation experiments (fivefold CV). The number of total samples is 12,000 and each fold has 2400 samples. For each run, the training data has 9600 samples (fourfolds) and test data has 2400 samples (onefold). The final error rates show the average of 5 runs.

Table 4 The error in the test data using ASIA

Target node	Visit to Asia	Tuberculousis	Smoking	Lung Cancer	Lung cancer or tuberculosis	Positive X-ray	Bronchitis	Dyspnoea	Total error
Single BN									
K2	0.26	0.05	2.66	0.02	*0.01*	*0.06*	5.24	9.12	17.46
GA	0.23	0.05	1.71	0.14	*0.01*	0.15	6.64	6.16	15.13
FSGA	0.23	0.05	1.75	0.05	*0.01*	0.15	5.51	*3.87*	11.65
DCGA	0.26	0.05	2.66	*0.01*	*0.01*	*0.06*	*5.15*	*3.87*	12.10
GA ensemble									
Expert	0.17	*0.04*	0.86	*0.01*	0.02	*0.06*	6.44	6.16	13.79
Greedy	0.17	0.05	1.00	0.14	*0.01*	0.06	6.40	6.16	14.03
$_{50}C_3$	0.18	0.05	1.10	0.14	*0.01*	*0.06*	6.42	6.16	14.15
FSGA ensemble									
Expert	*0.16*	*0.04*	*0.77*	0.04	*0.02*	*0.06*	5.22	*3.87*	*10.21*
Greedy	0.19	0.05	1.18	0.03	*0.01*	0.16	5.26	3.87	10.78
$_{50}C_3$	0.20	0.05	1.14	0.03	*0.01*	0.17	5.34	*3.87*	10.84
DCGA ensemble									
Expert	0.26	0.05	2.66	*0.01*	*0.01*	*0.06*	*5.15*	*3.87*	12.10
Greedy	0.26	0.05	2.66	*0.01*	*0.01*	*0.06*	*5.15*	*3.87*	12.10
$_{50}C_3$	0.26	0.05	2.66	*0.01*	*0.01*	*0.06*	*5.15*	*3.87*	12.10

Values in italics show the the lowest error of each column

Here we compare the prediction error of several algorithms for each target node. Three different types of evolutionary algorithm were used (GA, FSGA, and DCGA), and for each algorithm, three different ensemble selection methods were tested, i.e., combinatorial, greedy-style, and expert-style. K2 algorithm is a greedy-style topology search method

The "Single BN" with the best K2 metric was chosen from the GA, FSGA, and DCGA, respectively

Table 5 The error in the test data using ALARM

Target node	Anaphylaxis	Intubation status	Kinked ventilation tube	Disconnected ventilation tube	Hypovolemia	Left-ventricular failure	Insufficient anesthesia or analgesia	Pulmonary embolus	Total error
Single BN									
K2	0.57	8.57	12.19	*1.51*	1.29	*0.01*	*4.90*	*0.04*	29.29
GA	3.93	55.80	50.36	8.42	5.05	12.86	23.21	38.72	198.38
FSGA	3.85	21.18	41.42	13.33	5.60	10.90	22.66	53.63	172.60
DCGA	1.36	9.31	7.66	2.51	1.78	0.45	5.49	0.29	28.88
GA ensemble									
Expert	1.15	44.94	14.51	4.62	2.16	3.93	11.48	7.03	89.86
Greedy	2.03	51.91	18.81	6.26	4.67	7.94	14.55	17.57	123.77
$_{50}C_3$	2.45	49.05	22.16	5.90	5.20	7.40	15.83	20.36	128.40
FSGA ensemble									
Expert	1.14	17.82	9.29	3.60	2.01	1.23	8.30	8.43	51.85
Greedy	2.78	19.56	13.35	8.96	2.98	6.17	12.95	22.15	88.94
$_{50}C_3$	3.08	22.85	13.20	9.58	3.10	7.47	14.37	21.28	94.97
DCGA ensemble									
Expert	*0.06*	*8.38*	*7.36*	2.33	1.26	0.22	*4.90*	0.05	*24.60*
Greedy	0.28	8.58	7.52	1.70	*1.23*	0.27	5.07	0.19	24.88
$_{50}C_3$	0.32	8.58	7.56	1.72	1.29	0.30	5.14	0.32	25.26

Values in italics show the the lowest error of each column

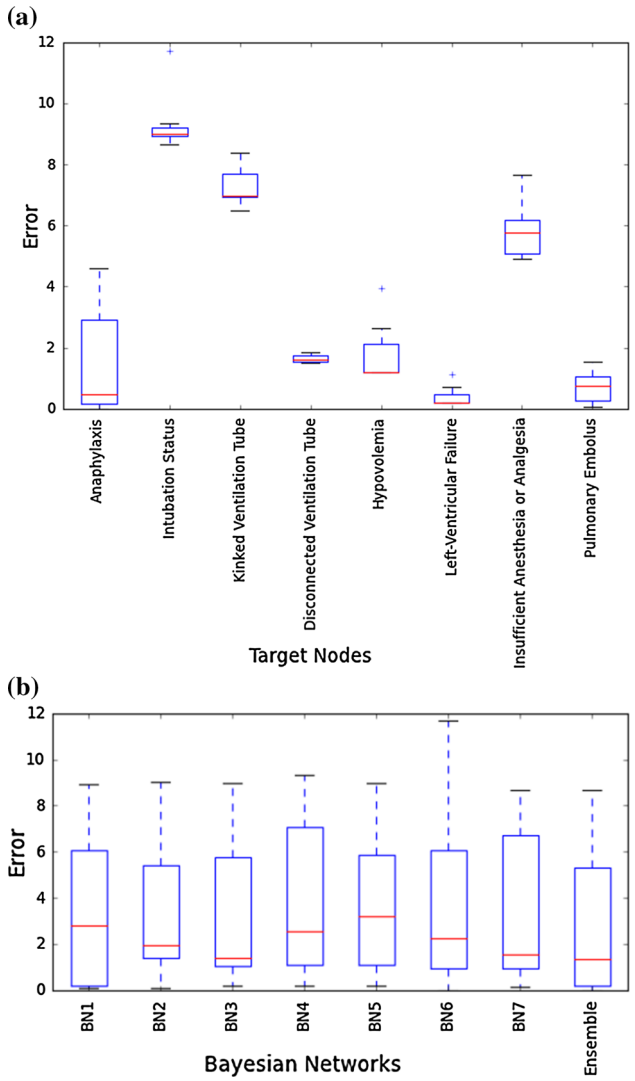

Fig. 7 An analysis of the members of the best ensemble for ALARM (DCGA+Expert). **a** The errors of the members in the ensemble for each node. **b** The sum of the errors for each target node of the members and the corresponding ensemble

In the results, they are similar to the one by 50 % training and 50 % testing experiments. In small networks, the SGA + Expert or FSGA + Expert are good choice. In large networks, the DCGA + Expert is recommended.

Figure 8 shows a comparison of the three heuristic ensemble search methods, i.e., Greedy, ${}_{50}C_3$ search and Expert. It shows that the expert-style performed better than the ${}_{50}C_3$ and greedy-style approaches. The analysis of ensemble size shows that the expert approach has more members than the greedy-style ensembles (Fig. 9). The ${}_{50}C_3$ ensembles always have three members. The results show that the ensemble size is related to the number of nodes in networks. Usually, the large networks require more members in the ensembles.

The time for the calculation is an important factor in choosing a learning algorithm. Because the evolutionary algorithm is a population-based method, it takes considerably longer than the greedy-style learning K2 algorithm, because it uses a local greedy search. Figure 10 shows a summary of the time for the calculations using GA. For small networks, it just takes 1–5 min to find the ensembles. For large networks, it takes about 2–4 h to complete all the process in the ensemble learning. One-half of time is used to train the Bayesian networks using evolutionary algorithms and the preparation stage of ensemble consumes another half.

The proposed algorithm consists of two steps: evolutionary learning (EL) and ensemble search (ES) steps. In the EL step, the calculation of the K2 metric for each BN in the population is the most time consuming part and the time complexity is proportional to the population size and the number of training cases. Because the K2 metric considers only the structure of BN, the conditional probability tables are not trained during the EL step. In terms of memory complexity, it is possible to reduce substantial memory requirement by storing only the non-zero entries. In the ES step, the time complexity is highly dependent on the "ensemble search strategy." At first, it trains the conditional probability table of all the individuals in the last generation. The ensemble search algorithm evaluates the goodness of each candidate, and the total time is dependent on the number of ensemble candidates and the number of training cases.

Table 8 summarizes the errors of the single and ensemble approaches for ASIA network with different population sizes. It shows that the error rate is very high if the population size is too small except DCGA. When the population size is 50, the GA, FSGA, and DCGA produce results with relatively low error rate. However, GA is still suffering from high standard deviation of errors. If the size is doubled, all three algorithms become stable with low deviation.

In Li et al. (2008) work, their goal was to combine multiple Bayesian networks to create a global BN. Following the approach, we tried to combine the topologies of the best three networks (based on the K2 metric) from the population of the last generation using three different operators (union, intersection, and voting). The union operator creates the edge in the global network if one of the source networks has it. On the other hand, the intersection operator creates an edge in the global network, only if all the source networks include the edge. Finally, the majority voting determines the edge based on the voting of source networks (Colace et al. 2014). The results show that the expert method outperforms the topological fusion in Asia and Alarm networks. In Alarm network, the expert method (24.60 ± 0.87) and intersect topological fusion (24.30 ± 0.51) have no statistically significant difference. Table 9 summarizes the results on the three combination methods.

Table 6 Summary of error rates on benchmark networks

Networks	Cancer	Earthquake	Survey	Asia	Insurance	Water	Alarm
Single BN							
K2	1.30	2.48	5.02	17.46	229.46	302.32	29.29
GA	1.30	2.48	5.02	15.13	1062.00	604.12	198.38
FSGA	61.60	2.53	12.06	11.65	1022.12	643.96	172.60
DCGA	1.30	2.48	5.02	12.10	215.28	302.32	28.88
GA ensemble							
Expert	*1.24*	*2.39*	*4.98*	13.79	477.64	304.47	89.86
Greedy	1.30	2.53	5.02	14.03	632.08	405.61	123.77
$_{50}C_3$	1.30	2.44	5.02	14.15	672.58	409.70	128.40
FSGA ensemble							
Expert	61.60	*2.39*	5.12	*10.21*	429.25	328.31	51.85
Greedy	61.60	2.52	5.78	10.78	657.29	412.28	88.94
$_{50}C_3$	61.60	2.47	6.25	10.84	701.85	425.42	94.97
DCGA ensemble							
Expert	1.30	2.48	5.02	12.10	215.25	*291.64*	*24.60*
Greedy	1.30	2.48	5.02	12.10	*212.95*	292.46	24.88
$_{50}C_3$	1.30	2.48	5.02	12.10	213.91	292.45	25.26

Values in italics show the the lowest error of each column

Table 7 Summary of cross-validation results (fivefold CV) on benchmark networks

Networks	Cancer	Earthquake	Survey	Asia	Water	Alarm (population size = 20)
Single BN						
K2	1.02	0.44	*1.23*	1.66	38.07	6.85
GA	1.02	0.44	*1.23*	1.79	191.26	95.46
FSGA	1.03	0.46	28.77	91.73	811.27	303.08
DCGA	1.05	0.48	1.31	1.63	37.27	7.03
GA ensemble						
Expert	1.02	0.44	*1.23*	*1.14*	91.79	69.93
Greedy	1.02	0.44	*1.23*	1.16	129.71	76.52
$_{50}C_3$	1.02	0.44	*1.23*	1.24	133.60	78.06
FSGA ensemble						
Expert	*1.00*	0.44	28.77	43.37	556.59	126.68
Greedy	1.02	*0.43*	28.77	69.45	694.48	176.88
$_{50}C_3$	1.03	*0.43*	28.77	92.26	724.18	195.50
DCGA ensemble						
Expert	1.05	0.48	1.31	1.63	36.42	*6.55*
Greedy	1.05	0.48	1.31	1.63	*36.31*	6.66
$_{50}C_3$	1.05	0.48	1.31	1.63	36.53	6.70

Values in italics show the the lowest error of each column

5 Conclusion

Speciation has been successfully applied to evolve multiple models and ensembles (Kim and Cho 2008; Kim and Cho 2005). Here, we applied these concepts to BNs. We compared the evolutionary algorithms GA, FSGA, and DCGA using three ensemble search heuristics, i.e., Expert, Greedy and $_{50}C_3$, with the seven benchmarking networks (small and large networks). The results show that the ensemble approach was more effective for generating predictive BNs than conventional greedy-style searches, such as K2, as well as single-network approaches.

In the evolutionary process, our contribution is to use speciation for the BN learning. Because the speciation focuses on both of accuracy and the diversity of population, it is not effective to find the fittest solution. However, the next ensem-

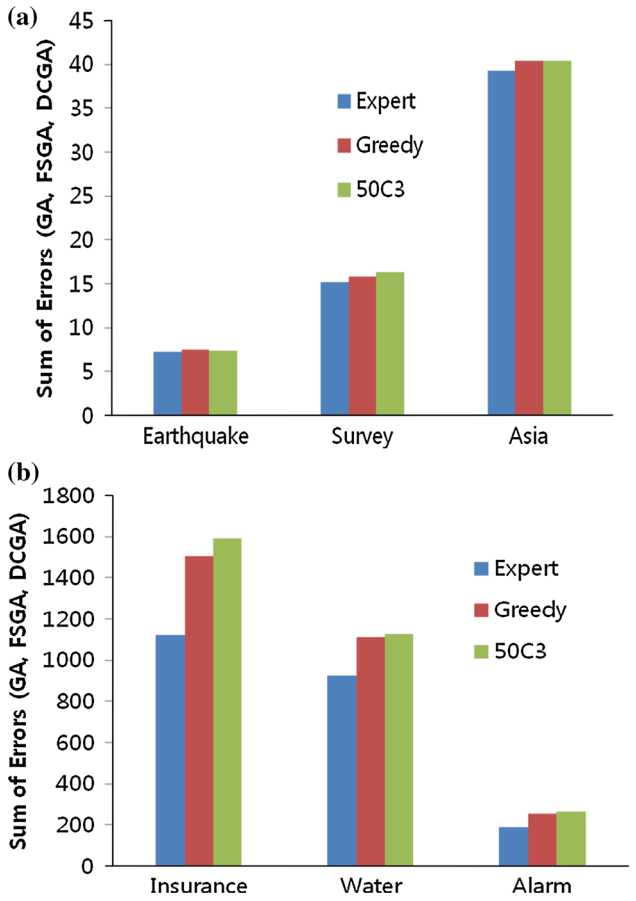

Fig. 8 A comparison of three heuristics for ensemble search summed over the three evolutionary algorithms. For each dataset, the errors in the three evolutionary algorithms were summed

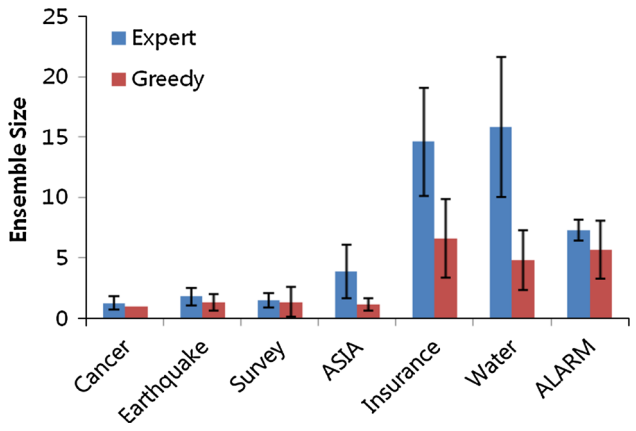

Fig. 9 The ensemble size (i.e., the number of BNs in the final ensemble) obtained using the expert and greedy search. The combinatorial heuristic assumes that an ensemble has three members; however, there is no restriction on the greedy-style heuristic. If the addition of new members is beneficial, it will continuously increase the number of members in the ensemble

ble search of our algorithm can be beneficial from the diverse solutions. Not all ensemble formation methods enumerate all the possible ensembles but only the ${}_{50}C_3$ technique does that.

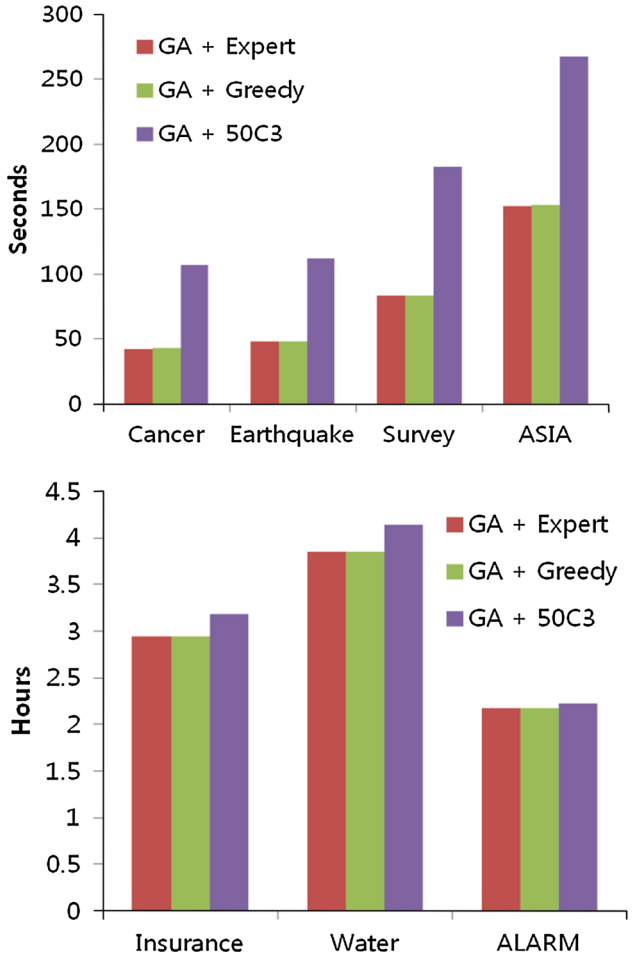

Fig. 10 The time for the calculations using a machine with an Intel i7 processor running at 3.20 GHz

The "expert" and "greedy" approaches attempt to select the subset of ensembles with less amount of time. In small-sized networks, the two approaches are about two times faster than the exhaustive one. However, in middle-sized networks, their time gap is not significant because training times is relatively big. Although the training time requires about 2–4 h for the middle-sized networks, they can be reduced with the parallelization techniques. If you apply the algorithm to different domains, you need to spend several hours depending on the size of networks to train the ensemble model. However, after the training, you do not need much time to get answers on unseen cases. It only depends on the inference algorithms' speed.

The time required to learn and find the ensembles was significant. There is considerable scope, however, to develop computational methods to reduce this time, such as multi-threaded programming, cashing, and approximate inference approaches. It is important to reduce the time required in the ensemble learning and the parallelization of evolutionary algorithms can be beneficial. The population-based search can be parallelized by distributing the tasks into multiple

Table 8 The impact of population size (Asia network)

ASIA network	POP_SIZE =10	POP_SIZE=50	POP_SIZE = 100
Single BN			
GA	102.77 ± 145.37	19.25 ± 14.00	12.33 ± 0.47
FSGA	2325.93 ± 1537.74	10.84 ± 0.84	11.57 ± 1.01
DCGA	*12.10 ± 0.00*	12.10 ± 0.00	12.10 ± 0.00
GA ensemble			
Expert	102.77 ± 145.37	17.02 ± 13.98	10.24 ± 0.76
Greedy	102.77 ± 145.37	17.48 ± 14.59	10.54 ± 0.78
$_{50}C_3$	102.77 ± 145.37	17.55 ± 14.66	10.54 ± 0.77
FSGA ensemble			
Expert	2325.93 ± 1537.74	*10.15 ± 0.60*	*10.07 ± 0.75*
Greedy	2325.93 ± 1537.74	10.76 ± 0.79	10.74 ± 0.75
$_{50}C_3$	2325.93 ± 1537.74	10.74 ± 0.71	17.50 ± 8.17
DCGA ensemble			
Expert	*12.10 ± 0.00*	12.10 ± 0.00	12.10 ± 0.00
Greedy	12.10 ± 0.00	*12.10 ± 0.00*	12.10 ± 0.00
$_{50}C_3$	12.10 ± 0.00	*12.10 ± 0.00*	12.10 ± 0.00
Average	813.6	19.25	11.77

Italic values show the lowest error of each column

Table 9 The results with topological fusion operators (ASIA Network, the combination of three best (based on K2 metric) networks from the population of the last generation)

	Expert method	Topological fusion		
		Union	Intersect	Majority voting Colace et al. (2014)
Asia				
GA	*13.79 ± 10.41*	15.20 ± 11.14	14.45 ± 11.52	14.61 ± 11.45
FSGA	*10.21 ± 0.45*	14.54 ± 2.70	27.93 ± 53.83	12.41 ± 1.55
DCGA	*12.10 ± 0.00*	*12.10 ± 0.00*	*12.10 ± 0.00*	*12.10 ± 0.00*
Alarm				
GA	*89.86 ± 43.95*	229.49 ± 119.38	289.80 ± 139.92	220.95 ± 161.88
FSGA	*51.85 ± 17.44*	190.99 ± 59.80	255.11 ± 152.25	167.20 ± 49.08
DCGA	*24.60 ± 0.87*	41.55 ± 3.74	*24.30 ± 0.51*	25.69 ± 1.07

Italic values show the lowest error of each column

threads (processes) on single or multiple machines. Parallelization has been used to expedite learning with BNs for applications in Big Data (Schadt et al. 2010) and large-scale networks (Vafaee 2014). In addition, instead of building a single BN from the large dataset, it is desirable to use distributed datasets with manageable sizes to learn using multiple BNs; "distributed Bayesian networks" (Na and Yang 2010) is an example of such an approach. We used the Bayesian score metrics to evaluate the goodness of the BN given the training samples. The time for the calculations increases exponentially with the number of samples because the metric enumerates all possible states to count their appearance in the dataset.

We used the Hamming distance to measure the similarity between BN structures; however, this distance measure is based only on the structure of the BNs. Furthermore, two BNs in the same class may have significant structural differences, while they can actually represent the same thing. A straightforward solution is to incorporate CPT into BNs and perform inferences on training datasets using this distance measurement. This will further increase the computational expense, and it is therefore desirable to develop more efficient methods to measure the similarity of BNs.

Acknowledgements This work was supported by the National Research Foundation of Korea (NRF) Grant funded by the Korean Government (MSIP) (2013 R1A2A2A01016589) and the Industrial Strategic Technology Development Program, 10044828, Development of augmenting multisensory technology for enhancing significant effect on service industry, funded by the Ministry of Trade, Industry & Energy (MI, Korea).

References

Barber D (2012) Bayesian reasoning and machine learning. Cambridge University Press, Cambridge

Beinlich IA, Suermondt HJ, Chavez RM, Cooper GF (1989) The ALARM monitoring system: a case study with two probabilistic inference techniques for belief networks. In: Proceedings of the Second European Conference on Artificial Intelligence in Medicine, pp 247–256

Binder J, Koller D, Russell S, Kanazawa K (1997) Adaptive probabilistic networks with hidden variables. Mach Learn 29(2–3):213–244

Chickering DM, Geiger D, Heckerman D (1994) Learning Bayesian networks is NP-hard, Technical Report MSR-TR-94-17, Microsoft Research

Colace F, De Santo M, Greco L (2014) Learning Bayesian network structure using a multiexpert approach. Int J Softw Eng Knowl Eng 24(2):269–284

Cooper GF, Herskovits E (1992) A Bayesian method for the induction of probabilistic networks from data. Mach Learn 9:309–347

Daly R, Shen Q, Aitken S (2011) Learning Bayesian networks: approaches and issues. Knowl Eng Rev 26(2):99–157

Feng G, Zhang J-D, Liao SS (2014) A novel method for combining Bayesian networks, theoretical analysis, and its applications. Pattern Recognit 47(5):2057–2069

Gamez JA, Mateo JL, Puerta JM (2011) Learning Bayesian networks by hill climbing: efficient methods based on progressive restriction of the neighborhood. Data Min Knowl Discov 22(1–2):106–148

Garg A, Pavlovic V, Rehg JM (2003) Boo sted learning in dynamic Bayesian networks for multimodal speaker detection. Proc IEEE 91(9):1355–1369

GENIE & SMILE. http://genie.sis.pitt.edu

Goldberg DE (2008) Genetic algorithms in search, optimization, and machine learning, 1st edn. Addison-Wesley Professional

Gouvea MM Jr., Araujo AFR (2010) Diversity-based adaptive evolutionary algorithms, Chapter 1. New Achievements in Evolutionary Computation

Heckerman D (2008) A tutorial on learning with Bayesian networks. Innov Bayesian Netw 156:33–82

Hu L, Wang L (2013) Using consensus Bayesian network to model the reactive oxygen species regulatory pathway. PLOS One 8(2):e56832. doi:10.1371/journal.pone.0056832

Hwang K-S, Cho S-B (2009) Landmark detection from mobile life log using a modular Bayesian network model. Expert Syst Appl 36:12065–12076

Jensen FV, Kjærulff U, Olesen KG, Pedersen J (1989) An expert system for control of waste water treatment—a pilot project. Technical report. Judex Datasystemer A/S, Aalborg (in Danish)

Kim K-J, Cho S-B (2005) Systematically incorporating domain-specific knowledge into evolutionary speciated checkers players. IEEE Trans Evol Comput 9(6):615–627

Kim K-J, Cho S-B (2008) Evolutionary ensemble of diverse artificial neural networks using speciation. Neurocomputing 71(7–9):1604–1618

Kim K-J, Cho S-B (2012) Automated synthesis of multiple analog circuits using evolutionary computation for redundancy-based fault-tolerance. Appl Soft Comput 12(4):1309–1321

Kim K, Mckay R (2012) Stochastic diversity loss and scalability in estimation of distribution genetic programming. IEEE Trans Evol Comput 17(3):301–320

Kim K-J, Park J-G, Cho S-B (2011) Correlation analysis and performance evaluation of distance measures for evolutionary neural networks. J Intell Fuzzy Syst 22:83–92

Kim KJ, Yoo JO, Cho SB (2005) Robust inference of Bayesian networks using speciated evolution and ensemble. In: International Symposium on Methodologies for Intelligent Systems, pp 92–101

Koller D, Friedman N (2009) Probabilistic graphical models: principles and techniques. MIT Press, Cambridge

Korb KB, Nicholson AE (2010) Bayesian artificial intelligence, 2nd edn. CRC Press, Boca Raton

Larranaga P, Karshenas H, Bielza C, Santana R (2013) A review on evolutionary algorithms in Bayesian network learning and inference tasks. Inf Sci 233(1):109–125

Larranaga P, Kuijpers CMH, Murga RH, Yurramendi Y (1996) Learning Bayesian network structures by searching for the best ordering with genetic algorithm. IEEE Trans Syst Man Cybern Part A 26(4):487–493

Larranaga P, Poza M, Yurramendi Y, Murga RH, Kuijpers CMH (1996) Structure learning of Bayesian networks by genetic algorithms: a performance analysis of control parameters. IEEE Trans Pattern Anal Mach Intell 18(9):912–926

Lauritzen S-L, Spiegelhalter DJ (1988) Local computations with probabilities on graphical structures and their applications on expert systems. J R Stat Soc B 50(2):157–224

Li XL, He XD, Yuan SM (2005) Learning Bayesian networks structures from incomplete data based on extending evolutionary programming. In: Proceedings of the Fourth International Conference on Machine Learning and Cybernetics, pp 2039–2043

Li W, Liu W, Yue K (2008) Recovering the global structure from multiple local Bayesian networks. Int J Artif Intell Tools 17(6):1067–1088

Luo X, Ouyang Y, Xiong Z (2011) Improving matrix factorization-based recommender via ensemble methods. Int J Inf Technol Decision Making 10(3):539–561

Mahfoud SW (1995) Niching methods for genetic algorithms. Ph.D. Dissertation, University of Illinois at Urbana-Champaign

Muruzabal J, Cotta C (2007) A Study on the evolution of Bayesian network graph structures. Adv Probab Graph Models 193–214

Myers JW, Laskey KB, Dejong KA (1999) Learning Bayesian networks from incomplete data using evolutionary algorithm. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp 458–465

Na Y, Yang J (2010) Distributed Bayesian network structure learning. In: IEEE International Symposium on Industrial Electronics, pp 1607–161

Pena JM, Kocka T, Nielsen JD (2004) Featuring multiple local optima to assist the user in the interpretation of induced Bayesian network models. In: Proceedings of the Tenth International Conference on Information Processing and Management of Uncertainty in Knowledge-based Systems, pp 1683–1690

Peng Y, Kou G, Wang G, Wu W, Shi Y (2011) Ensemble of software defect predictors: an AHP-based evaluation method. Int J Inf Technol Decision Making 10(1):187–206

Robles V, Larranaga P, Pena JM, Menasalvas E, Perez MS, Herves V, Wasilewska A (2004) Bayesian network multi-classifiers for protein secondary structure prediction. Artif Intell Med 31(2):117–136

Rogers A, Prugel-Bennett A (1999) Genetic drift in genetic algorithm selection schemes. IEEE Trans Evol Comput 3(4):298–303

Schadt EE, Linderman MD, Sorenson J, Lee L, Nolan GP (2010) Computational solutions to large-scale data management and analysis. Nat Rev Genet 11(9):647–657

Scutari M, Denis JB (2014) Bayesian networks: with examples in R. Chapman & Hall, London

Shen C-W (2009) A Bayesian networks approach to modeling financial risks of e-logistics investments. Int J Inf Technol Decision Making 8(4):711–726

Su X, Khoshgoftaar TM (2008) Collaborative filtering for multi-class data using Bayesian networks. Int J Artif Intell Tools 17(1):71–85

Vafaee F (2014) Learning the structure of large-scale Bayesian networks using genetic algorithm. In: Proceedings of the Conference on Genetic and Evolutionary Computation, pp 855–862

Wong ML, Lam W, Leung KS (1999) Using evolutionary programming and minimum description length principle for data mining of Bayesian networks. IEEE Trans Pattern Anal Mach Intell 21(2):174–178

Wong ML, Lee SY, Leung KS (2004) Data mining of Bayesian networks using cooperative coevolution. Decision Support Syst 38:451–472

Zhou ZH (2012) Ensemble methods: foundations and algorithms. Chapman & Hall/CRC, London