International Journal of Control, Automation, and Systems (2010) 8(2):398-407 http://www.springer.com/12555
DOI 10.1007/s12555-010-0227-3

A New Genetic Approach for Structure Learning of Bayesian Networks: Matrix Genetic Algorithm

Jaehun Lee, Wooyong Chung, Euntai Kim*, and Soohan Kim

Abstract: In this paper, a novel method for structure learning of a Bayesian network (BN) is developed. A new genetic approach called the matrix genetic algorithm (MGA) is proposed. In this method, an individual structure is represented as a matrix chromosome and each matrix chromosome is encoded as concatenation of upper and lower triangular parts. The two triangular parts denote the connection in the BN structure. Further, new genetic operators are developed to implement the MGA. The genetic operators are closed in the set of the directed acyclic graph (DAG). Finally, the proposed scheme is applied to real world and benchmark applications, and its effectiveness is demonstrated through computer simulation.

Keywords: Bayesian network, connectivity matrix, genetic algorithm, matrix chromosome, structure learning.

1. INTRODUCTION

A Bayesian network (BN) is a probabilistic approach for reasoning under uncertainty, and has become a popular knowledge representation scheme in several fields such as data mining and knowledge discovery [1-3]. A BN is a graphical model which denotes a joint probabilistic distribution of given variables under their dependence relationships. In a BN, structure is bounded to a directed acyclic graph (DAG). Each node is connected to its parent's nodes and each arc represents the conditional dependency between two connected nodes.

When a database of cases is provided and a BN is to be designed, the most important yet difficult job is to determine the structure of the BN. For example, one must determine how many nodes to use and which node should be connected to which node (that is, the topology of the BN). This task is called the structure learning of the BN, and several methods for this have been reported [4-11]. The most popular method among these is the K2 algorithm [11]. The K2 algorithm assumes ordering among variables and searches for an appropriate structure under the given ordering. But this method is heuristic, and its performance is not guaranteed. In addition, if no or only partial information is available about the causal relationship among the variables, the K2 algorithm does not work well.

Genetic Algorithms (GAs) are an alternative scheme that can be applied to the structure learning problem. They provide an adaptive and robust optimization procedure, and are an effective choice for the structure learning of the BN, because learning is highly combinatorial in a huge search space. Several studies based on GAs have been reported [7-9]. However, most of the existing methods predetermine the order of BN variables thereby restricting the search space of the BN structure.

In this paper, a new structure learning method for BN is developed. We propose a matrix chromosome to represent the BN structure. Each chromosome is decomposed into two triangular parts: the first triangular part represents the conditional dependencies among the BN nodes in a fixed order, and the second triangular part represents the conditional dependencies among the BN nodes in reverse order. This allows all possible connections among BN nodes, unlike existing methods.

The remainder of the paper is organized as follows. In Section 2, a brief introduction to the BN and GAs is given. In Section 3, a new algorithm for the structure learning of the BN is proposed, and the appropriate genetic operators are introduced. In Section 4, the proposed method is applied to real world and benchmark problems including a database of virtual home network systems [14], a database of car diagnosis problems as suggested by Norsys [15], and a database of ALARM networks constructed by Beinlinch *et al.* [11]. Finally, some conclusions are drawn in Section 5.

2. PRELIMINARY FUNDAMENTALS: BAYESIAN NETWORKS AND GENETIC ALGORITHMS

2.1. Bayesian networks

A Bayesian network (BN) is a probabilistic framework

Manuscript received November 20, 2008; accepted September 27, 2009. Recommended by Editorial Board member Sungshin Kim under the direction of Editor Young-Hoon Joo. This work was supported by the Ministry of Commerce, Industry and Energy of Korea (HISP). E. Kim appreciates the financial support from LG Yonam Foundation during his sabbatical year at the University of California, Berkeley.

Jaehun Lee, Wooyong Chung, and Euntai Kim are with the School of Electrical and Electronic Engineering, Yonsei University, 134, Sinchon-dong, Seodaemun-gu, Seoul 120-749, Korea (e-mails: {aznable17,wychung, etkim}@yonsei.ac.kr).

Soohan Kim is with the Network Sys. Div. Internet Infra Team, Samsung Electronics, 416, Matan-3Dong Suwon 443-742, Korea (e-mail: ksoohan@samsung.com).

* Corresponding author.

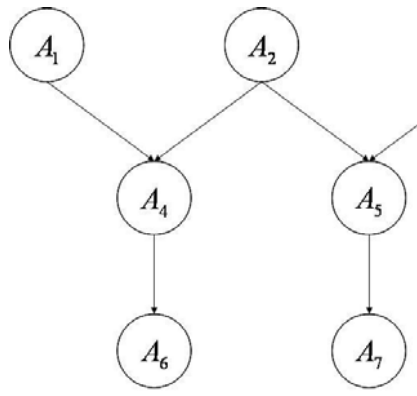

Fig. 1. An example of a bayesian network.

for reasoning under uncertainty, and has gained popularity in recent years in the artificial intelligence community. A BN is composed of a network structure and a set of parameters associated with the structure. The structure of the BN is a directed acyclic graph (DAG), in which each node represents a random variable and each arc represents the conditional dependency (topology) between two nodes [16].

To specify the BN completely with respect to the DAG, the prior probabilities for all root nodes (nodes with no predecessors) and conditional probabilities for all other nodes must be determined. The joint probability of any particular instantiation of all variables in the BN can be calculated by

$$P(U) = \prod_i P\left(A_i \mid pa(A_i)\right), \tag{1}$$

where $U = \{A_1, A_2, \cdots, A_n\}$ is a set of nodes of the BN, n is the number of variables, and $pa(A_i)$ is the parent set of the variable A_i. For example, the probability of the BN shown in Fig. 1 is computed by

$$\begin{aligned} P(U) = {} & P(A_1)P(A_2)P(A_3)P(A_4 \mid A_1, A_2) \\ & P(A_5 \mid A_2, A_3)P(A_6 \mid A_4)P(A_7 \mid A_5). \end{aligned} \tag{2}$$

The process of building the BN can be separated into two tasks: structure learning and parameter learning. Structure learning involves determining an appropriate structure for the BN such that the BN accommodates the given set of samples. Parameter learning includes computing the conditional probabilities for the given BN structure such that the output of the BN approximates the distribution of the given set of samples. The most popular parameter learning method is the expectation maximization (EM) algorithm [17]. In this paper, we focus on the structure learning of the BN and build an appropriate BN structure.

2.2. Genetic algorithm

A genetic algorithm (GA) is the implementation of a biological metaphor. In the algorithm, learning is viewed as a competition among a population of evolving individuals. The goodness of each candidate solution is evaluated based on its fitness and the population evolves

```
procedure SGA()
    initialize(Population);
    evaluate(Population);
    while not (terminal condition satisfied) do
        MatingPool = reproduce(Population);
        MutationPool = crossover(MatingPool);
        Population = mutation(MutationPool);
        evaluate(Population);
    end
end
```

Fig. 2. The pseudo code of a simple genetic algorithm.

by selection, crossover, and mutation.

The general procedure for a GA is summarized as follows. An initial population of individuals called chromosomes is randomly created. Then, promising individuals are selected to reproduce offspring for the next generation. The number of copies in a generation is proportional to their relative fitness value. The selected individuals undergo crossover and mutation to search for a global optimal solution. When the new population includes an individual that has a satisfactory fitness value, the algorithm stops and the problem is solved. If not, then the algorithm is repeated until a termination condition is satisfied. The pseudo code of a simple genetic algorithm (SGA) is shown in Fig. 2.

3. NEW STRUCTURE LEARNING OF BAYESIAN NETWORK BASED ON GENETIC ALGORITHMS

Over the past decade, GAs have been considered as the promising structure learning method for BN compared to K2, since K2 is too heuristic. In this section, a new approach for structure learning of a BN, based on a genetic algorithm, is proposed. The proposed method explores a wider solution space than the existing GA-based structure learning methods.

3.1. Existing methods for BN structure learning

In this subsection, existing methods [7,10,19] for structure learning of BNs based on GAs are briefly summarized. The BN structure with n variables is represented by a $n \times n$ connectivity matrix $C = (c_{ij})$, where

$$c_{ij} = \begin{cases} 1 & \textit{if i is a parent of j} \\ 0 & \textit{otherwise}, \end{cases} \tag{3}$$

and each BN is encoded as a chromosome,

$$c_{11}c_{12} \cdots c_{1n}c_{21}c_{22} \cdots c_{2n} \cdots c_{n1}c_{n2} \cdots c_{nn}. \tag{4}$$

With this representation, the plain crossover and mutation would produce illegal (non-DAG) BN structures. In [7], to overcome this problem, the connectivity matrix was limited to being upper triangular as

$$\begin{pmatrix} 0 & c_{12} & c_{13} & \cdots & c_{1n-1} & c_{1n} \\ 0 & 0 & c_{23} & \cdots & c_{2n-1} & c_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & c_{n-1n} \\ 0 & 0 & 0 & \cdots & 0 & 0 \end{pmatrix}, \tag{5}$$

and the connectivity matrix was encoded as a chromosome,

$$X = c_{12}c_{13}\cdots c_{1n}c_{23}c_{24}\cdots c_{2n}\cdots c_{n-2n-1}c_{n-2n}c_{n-1n}. \tag{6}$$

In other words, the ordering among the BN variables was fixed and node A_i was allowed to have another node A_j as a parent node only if node A_j comes before node A_i in the ordering. This scheme restricts c_{ij} to 0 and narrows the search space such that the GA can find only a suboptimal solution if the fixed ordering of the nodes is wrong. If there is no prior knowledge about the ordering between variables, as is usually the case, the resulting structure will not be satisfactory.

To overcome this limitation, the ordering and connectivity among variables are encoded separately and optimized simultaneously in [10] and [19]. In [10], the ordering and connectivity among the variables are encoded as

$$\begin{aligned} X_o &= x_1x_2\cdots x_n, \\ X_c &= c_{12}c_{13}\cdots c_{1n}c_{23}c_{24}\cdots c_{2n}\cdots c_{n-2n-1}c_{n-2n}c_{n-1n}, \end{aligned} \tag{7}$$

where X_o is an integer-coded chromosome and denotes the ordering among the BN nodes, $x_i \in \{1,\cdots,n\}$ is an integer, and $x_i \neq x_j$ *iff* $i \neq j$. X_c is a binary-coded chromosome and denotes the upper triangular connectivity matrix by

$$C = \begin{pmatrix} 0 & c_{12} & c_{13} & \cdots & c_{1n-1} & c_{1n} \\ 0 & 0 & c_{23} & \cdots & c_{2n-1} & c_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & c_{n-1n} \\ 0 & 0 & 0 & \cdots & 0 & 0 \end{pmatrix}, \tag{8}$$

$$c_{ij} = \begin{cases} 1 & \textit{if } x_i \textit{ is a parent of } x_j \\ 0 & \textit{otherwise.} \end{cases}$$

Further, it was rigorously proven in [10] that there is no BN structure that cannot be represented by this dual encoding, and that this dual encoding explores the entire solution space of BN structures.

3.2. Matrix genetic algorithm for BN structure learning

In [10], the entire solution space was searched for the fittest BN structure. However, the limitation of this is that when crossover is applied to connectivity chromosomes, the resulting offspring BNs may completely or at least seriously differ from the parent BNs because they have different ordering chromosomes. Thus, it is very likely that superior features of the parents are not inherited in the offspring, degrading the performance of evolution.

Thus, to overcome this problem, we propose a new genetic method called the *Matrix Genetic Algorithm* (MGA) for structure learning of the BN. Unlike existing methods, the ordering between nodes is fixed, but the connectivity matrix is not confined to being triangular. Instead, the BN structure is encoded as a matrix chromosome composed of upper and lower triangular matrices. We also introduce new genetic crossover and mutation operators tailored for the MGA.

3.2.1 Encoding

We encode each BN structure with variables as follows:

$$\begin{aligned} X &= X_uX_l, \\ &= c_{12}c_{13}\cdots c_{1n}c_{23}\cdots c_{2n}\cdots c_{n-1n}c_{21}c_{31}c_{32}\cdots c_{nn-1}, \end{aligned} \tag{9}$$

where

$$C = \begin{pmatrix} 0 & c_{12} & c_{13} & \cdots & c_{1n-1} & c_{1n} \\ c_{21} & 0 & c_{23} & \cdots & c_{2n-1} & c_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ c_{n-11} & c_{n-12} & \cdots & c_{n-1n-2} & 0 & c_{n-1n} \\ c_{n1} & c_{n2} & \cdots & c_{nn-2} & c_{nn-1} & 0 \end{pmatrix}, \tag{10}$$

$$c_{ij} = \begin{cases} 1 & \textit{if } x_i \textit{ is a parent of } x_j \\ 0 & \textit{otherwise,} \end{cases}$$

$$\begin{aligned} X_u &= c_{12}c_{13}\cdots c_{1n}c_{23}\cdots c_{2n}\cdots c_{n-1n}, \\ X_l &= c_{21}c_{31}c_{32}c_{41}\cdots c_{43}\cdots c_{n1}\cdots c_{nn-1}. \end{aligned}$$

It should be noted that, unlike existing methods [7,10,19], the connectivity of the BN structure is encoded not as an upper triangular matrix but as a full matrix with zeros on the diagonal. *Thus, this encoding intuitively encompasses all the possible BN structures but could include the cycle, which is a violation of the acyclic property of the BN.*

3.2.2 Crossover

Here, we introduce a new crossover operator for the proposed MGA. First, the crossover operation is applied to the site between the upper and lower sub-chromosomes (triangular matrices). The crossover at the site between the upper and lower sub-chromosomes may violate the acyclic property of the BN and, if so, we repair the offspring simply by removing their cycles. More specifically, let us consider the following two BN structures:

$$\begin{aligned} X_1 &= X_{u_1}X_{l_1}, \\ X_2 &= X_{u_2}X_{l_2}, \end{aligned} \tag{11}$$

where

$$\begin{aligned} X_{u_1} &= c^1_{12}c^1_{13}\cdots c^1_{1n}c^1_{23}\cdots c^1_{2n}\cdots c^1_{n-1n}, \\ X_{l_1} &= c^1_{21}c^1_{31}c^1_{32}c^1_{41}\cdots c^1_{43}\cdots c^1_{n1}\cdots c^1_{nn-1}, \end{aligned}$$

$$
\begin{aligned}
X_{u_2} &= c_{12}^2 c_{13}^2 \cdots c_{1n}^2 c_{23}^2 \cdots c_{2n}^2 \cdots c_{n-1n}^2, \\
X_{l_2} &= c_{21}^2 c_{31}^2 c_{32}^2 c_{41}^2 \cdots c_{43}^2 \cdots c_{n1}^2 \cdots c_{nn-1}^2.
\end{aligned} \tag{12}
$$

After the first crossover operation, the resulting offspring chromosomes become

$$
\begin{aligned}
X_1^c &= X_{u_1} X_{l_2}^r \ \ or \ \ X_{u_1}^r X_{l_2}, \\
X_2^c &= X_{u_2} X_{l_1}^r \ \ or \ \ X_{u_2}^r X_{l_1},
\end{aligned} \tag{13}
$$

where

$$
\begin{aligned}
X_{u_k}^r &= c_{12} c_{13} \cdots c_{1n} c_{23} c_{24} \cdots c_{2n} \cdots c_{(n-1)n}, \\
X_{l_k}^r &= c_{21} c_{31} c_{32} c_{41} \cdots c_{43} \cdots c_{n1} c_{n2} \cdots c_{n(n-1)}, \\
c_{ij} &= \begin{cases} 0 & if \ a \ cycle \ occurs \ inside \ the \ network \\ c_{ij}^k & otherwise. \end{cases}
\end{aligned} \tag{14}
$$

That is, $X_{l_2}^r$ is the alteration of X_{l_2} obtained by simply removing the inner cycle in the chromosome $X_{u_1} X_{l_2}$. In this crossover, there are two possible options for each offspring. For example, for X_1^c, the options are $X_{u_1} X_{l_2}^r$ or $X_{u_1}^r X_{l_2}$ and either of the two chromosomes are selected according to the following probabilities.

For X_1^c,

$$
\begin{aligned}
P(X_1^c = X_{u_1} X_{l_2}^r) &= \frac{\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} c_{ij}^{\ 1}}{\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} c_{ij}^{\ 1} + \sum_{i=2}^{n} \sum_{j=1}^{i-1} c_{ij}^{\ 2}}, \\
P(X_1^c = X_{u_1}^r X_{l_2}) &= \frac{\sum_{i=2}^{n} \sum_{j=1}^{i-1} c_{ij}^{\ 2}}{\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} c_{ij}^{\ 1} + \sum_{i=2}^{n} \sum_{j=1}^{i-1} c_{ij}^{\ 2}},
\end{aligned} \tag{15a}
$$

and for X_2^c,

$$
\begin{aligned}
P(X_2^c = X_{u_2} X_{l_1}^r) &= \frac{\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} c_{ij}^{\ 2}}{\sum_{i=2}^{n} \sum_{j=1}^{i-1} c_{ij}^{\ 1} + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} c_{ij}^{\ 2}}, \\
P(X_2^c = X_{u_2}^r X_{l_1}) &= \frac{\sum_{i=2}^{n} \sum_{j=1}^{i-1} c_{ij}^{\ 1}}{\sum_{i=2}^{n} \sum_{j=1}^{i-1} c_{ij}^{\ 1} + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} c_{ij}^{\ 2}}.
\end{aligned} \tag{15b}
$$

The logic for this choice is that a triangle with more connections as a dominant half is selected and the other triangle with fewer connections after repair is connected such that the inner cycle is removed. The choice, however, is not deterministic but probabilistic as in equation (15). For example, for X_1^c, the sub-chromosome X_{u_1} is selected for the first half of the chromosome (the upper triangular part) with a probability from equation (15) and if so, X_{l_2} is modified (or repaired) into $X_{l_2}^r$ by (14) and X_{u_1} and $X_{l_2}^r$ are concatenated into a single chromosome.

The second crossover operation is applied to the inside of the upper and lower sub-chromosomes. Consider the two BN structures in equation (11) and let us index the subchromosomes as

$$
\begin{aligned}
X_{u_i} &= u_1^{\ i} u_2^{\ i} \cdots u_{\frac{n(n-1)}{2}}^{\ \ i}, \\
X_{l_i} &= l_1^{\ i} l_2^{\ i} \cdots l_{\frac{n(n-1)}{2}}^{\ \ i} \quad \text{where} \quad i = 1, 2
\end{aligned} \tag{16}
$$

for convenience of explanation. If the second crossover occurs between the ith and $i+1$th genes in the first half sub-chromosome and the second crossover occurs between the jth and $j+1$th genes in the second half sub-chromosome, then the second crossover is defined as

$$
\begin{aligned}
X_1^c &= X_{u_1^c} X_{l_1^c}^r, \\
X_2^c &= X_{u_2^c}^r X_{l_2^c},
\end{aligned} \tag{17}
$$

where

$$
\begin{aligned}
X_{u_1^c} &= u_1^{\ 1} u_2^{\ 1} \cdots u_i^{\ 1} u_{i+1}^{\ 2} u_{i+2}^{\ 2} \cdots u_{\frac{n(n-1)}{2}}^{\ \ 2}, \\
X_{l_1^c} &= l_1^{\ 1} l_2^{\ 1} \cdots l_j^{\ 1} l_{j+1}^{\ 2} l_{j+2}^{\ 2} \cdots l_{\frac{n(n-1)}{2}}^{\ \ 2}, \\
X_{u_2^c} &= u_1^{\ 2} u_2^{\ 2} \cdots u_i^{\ 2} u_{i+1}^{\ 1} u_{i+2}^{\ 1} \cdots u_{\frac{n(n-1)}{2}}^{\ \ 1}, \\
X_{l_2^c} &= l_1^{\ 2} l_2^{\ 2} \cdots l_j^{\ 2} l_{j+1}^{\ 1} l_{j+2}^{\ 1} \cdots l_{\frac{n(n-1)}{2}}^{\ \ 1},
\end{aligned} \tag{18}
$$

and $X_{l_1^c}^r$, $X_{u_2^c}^r$ is the repaired result of $X_{l_1^c}$, $X_{u_2^c}$ using equation (14) such that there are no cycles in the corresponding chromosome.

3.2.3 Mutation

In this subsection, we develop a new mutation operator for the proposed MGA. As in the crossover operations, we select a triangle with more connections as a dominant half and mutate the other triangle with fewer connections randomly such that there is no cycle in the resulting chromosome. More specifically, if $X = X_u X_l$ is mutated, the new chromosome is

$$
X^m = \begin{cases} X_u X_{l^m}^r & if \ \ \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} c_{ij} > \sum_{i=2}^{n} \sum_{j=1}^{i-1} c_{ij} \\ X_{u^m}^r X_l & otherwise, \end{cases} \tag{19}
$$

where $X_{l^m}^r$ and $X_{u^m}^r$ are random but repaired by (14) such that there is no cycle in $X_u X_{l^m}^r$ or $X_{u^m}^r X_l$. Therefore,

$$X_{u^m}^r = c_{12}^r c_{13}^r \cdots c_{1n}^r c_{23}^r c_{24}^r \cdots c_{2n}^r \cdots c_{(n-1)n}^r,$$
$$X_{l^m}^r = c_{21}^r c_{31}^r c_{32}^r c_{41}^r \cdots c_{43}^r \cdots c_{n1}^r c_{n2}^r \cdots c_{n(n-1)}^r,$$
$$c_{ij}^r = \begin{cases} 0 & \textit{if a cycle occurs inside the network} \\ 0 \textit{ or } 1 & \textit{otherwise.} \end{cases} \tag{20}$$

3.2.4 Fitness function

To evaluate the given chromosome (the BN structure), we use the following theorem which was reported in [11].

Theorem 1: Let U be a set of n discrete variables where a variable A_i in U has r_i possible value assignments: $(v_{i1},\cdots,v_{ir_i})$. Let D be a database of m cases where each case contains a value assignment for each variable in U. Let B_s denote a belief network structure containing just the variables in U. Each variable A_i in B_s has a set of parents which are represented with a list of variables $pa(A_i)$. Let w_{ij} denotes the j th unique instantiation of $pa(A_i)$ relative to D. Suppose there are q_i such unique instantiations of $pa(A_i)$. Define N_{ijk} to be the number of cases in D in which variable A_i has the value v_{ik} and $pa(A_i)$ is instantiated as w_{ij}. Let $N_{ij} = \sum_{k=1}^{r_i} N_{ijk}$. If the cases occur independently and the probability density function, $f(B_p \mid B_s)$, is uniform, then it follows that

$$P(B_s, D) = P(B_s) \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{(r_i-1)!}{(N_{ij}+r_i-1)!} \prod_{k=1}^{r_i} N_{ijk}! \tag{21}$$

Since $P(B_s, D) = P(B_s)P(D \mid B_s)$ and $P(B_s)$ is uniformly distributed, the fittest BN structure B_s is determined by maximizing

$$P(D \mid B_s) = \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{(r_i-1)!}{(N_{ij}+r_i-1)!} \prod_{k=1}^{r_i} N_{ijk}! \tag{22}$$

4. SIMULATION

In this section, the proposed method is applied to three real world problems and its performance is compared with those of existing methods. The three problems considered are the *home network system* [14, 20*], the car diagnosis problem* [15], and the *ALARM network* [21].

For each problem, a fixed set of GA parameters was used including a population size (P_{size}) of 50, a crossover rate (p_c) of 0.65 and a mutation rate (p_m) of 0.05. The three methods were then compared in terms of the effectiveness of the structure. The same simulation was then repeated while varying the GA parameters and checking whether the comparison results were consistent. Two population sizes (50 and 100), three crossover rates (0.4, 0.65, and 0.9), and two mutation rates (0.05 and 0.15) were used. For each set of GA parameters, 10 independent runs were made with each run stopped when 5,000 BN structures were evaluated.

In all runs, the maximum number of parents of each node (in-degree of BN structure) was restricted to five for simplicity. Without this restriction, the number of possible configurations of the parent set would increase exponentially with the number of parents [22].

4.1. Home network system

A home network system is a typical test bed of the ubiquitous computing and sensor network, and is expected to upgrade the quality of living over the next decade. In a home network system, the key issue is context-aware computing. Context-aware computing is jargon coined by computer scientists and it is aimed at providing users in a smart home with human friendly services by (1) gathering information about the users from various sensors distributed in a smart home and (2) recognizing the intention of the users.

In the implementation of context-aware computing in a smart home, it is assumed that home appliances and devices are equipped with wired/wireless sensors, and information about the circumstances under which the appliances operate are gathered. It is known that the contexts of user activities can be presumed from the state of the environment in [20], but there is no well known tool for developing the context-aware applications.

In this example, the BN is applied to the context-aware computing. We developed a virtual home in which a single user lives and home appliances are equipped with wired/wireless sensors. His (or her) intention or action is presumed by the BN. A web-based virtual action simulator was developed in FLASH and a database for a context-aware system was built to train the BN [14]. Fig. 3 shows the BN structure of the context-aware system.

Forty-two random variables were used to implement the context awareness of the BN. Twenty-one variables represent the state of the home appliances such as the TV, lights in the living room, the refrigerator, among others. The other twenty-one variables represent the activities of a single user such as whether he/she is studying, sleeping, or dressing. Using the proposed method, the BN is

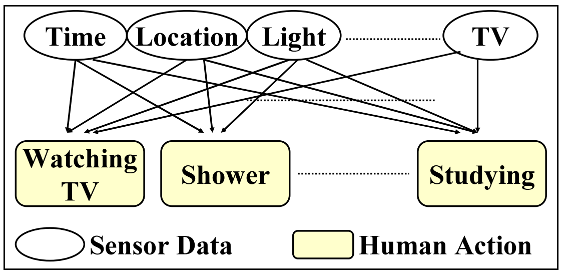

Fig. 3. An example of the BN structure for a home network system.

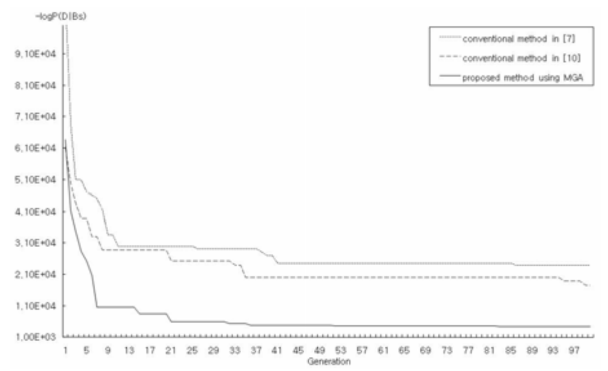

Fig. 4. Performance of the existing and proposed methods (home network system).

trained such that the action of the user is presumed from measurements of the sensors distributed in the context-aware home.

Fig. 4 shows the structure learning result of a specific run under the standard set of GA parameters (population size = 50, crossover rate = 0.65, mutation rate = 0.05). The short-dashed, long-dashed, and solid lines denote the performances of the methods used in [7,10], and the MGA, respectively. In the figure, a lower vertical value indicates better performance since the negative logarithm of the probability is plotted.

As the vertical value approaches zero, the probability approaches one meaning that BN accommodates all the cases in the database. From Fig, 4, it can be seen that the proposed method converges quickly and outperforms the existing methods [7,10]. The reasons for the superiority of the proposed method could be (1) no prior knowledge of the causal relation for the given network is assumed and, as stated in Section 3, (2) the proposed MGA passes the superior genes from generation to generation more efficiently than the method in [10].

Table 1 compares results from the methods in [7,10], and the proposed method in terms of equation (22). The proposed MGA outperforms the existing methods not only on average, but also in the best and worst cases.

Tables 2, 3, and 4 show the performances of the methods in [7,10], and the proposed method in terms of equation (22), respectively, while varying the GA parameters. The set of GA parameters used in the simulation are listed in the first column. From the tables, it can be seen that all the three methods perform their

Table 1. The comparison of the proposed and existing methods (home network system).

Trial time	Larrañaga *et al.* [7]	Lee *et al.* [10]	Proposed MGA
1	23895.26	17626.14	4460.89
2	28117.36	21105.52	4481.27
3	21809.5	20482.32	4458.11
4	19619.23	20547.76	4458.11
5	21850.03	22245.77	4458.11
6	21608.97	19160.47	4458.11
7	18404.05	22117.03	4458.11
8	21460.08	21836.7	4482.56
9	23702.79	21367.96	4458.11
10	24606.22	20128.63	4458.11
avg.	22507.35	20661.83	4463.15
var.	7470452.4	2055700.7	98.67
Best	18404.05	17626.14	4458.11
worst	28117.36	22245.77	4481.27

Table 2. The performances of the method in [7] with various parameters (home network system).

GA parameters	Average	Variance
	Best	Worst
$p_c = 0.4, p_m = 0.05, P_{size} = 50$	33590.13	7848290.10
	29545.27	38091.62
$p_c = 0.9, p_m = 0.05, P_{size} = 50$	33974.26	3413800.24
	31705.53	36576.58
$p_c = 0.4, p_m = 0.15, P_{size} = 50$	45338.2	3355819.07
	42070.51	47054.68
$p_c = 0.65, p_m = 0.15, P_{size} = 50$	46324.2	4205066.25
	43866.84	49754.24
$p_c = 0.9, p_m = 0.15, P_{size} = 50$	45199.51	9951885.5
	39930.35	48574.1
$p_c = 0.65, p_m = 0.05, P_{size} = 50$	22507.35	7470452.42
	18404.05	28117.36
$p_c = 0.65, p_m = 0.05, P_{size} = 100$	33405.87	5500013.51
	26880.32	36082.38

Table 3. The performances of the method in [10] with various parameters (home network system).

GA parameters	Average	Variance
	Best	Worst
$p_c = 0.4, p_m = 0.05, P_{size} = 50$	23956.98	2517574.58
	21653.24	25930.32
$p_c = 0.9, p_m = 0.05, P_{size} = 50$	21337.34	1367195.86
	19927.22	23461.91
$p_c = 0.4, p_m = 0.15, P_{size} = 50$	24292.19	7288403.13
	20650.62	28602.93
$p_c = 0.65, p_m = 0.15, P_{size} = 50$	23200.76	3425767.05
	20421.56	25839.42
$p_c = 0.9, p_m = 0.15, P_{size} = 50$	26023.81	9070632.76
	22740.04	30872.69
$p_c = 0.65, p_m = 0.05, P_{size} = 50$	20661.83	2055700.73
	17626.14	22245.77
$p_c = 0.65, p_m = 0.05, P_{size} = 100$	22518.89	3728110.08

Table 4. The performances of the proposed MGA with various parameters (home network system).

GA parameters	Average	Variance
	Best	Worst
$p_c = 0.4, p_m = 0.05, P_{size} = 50$	4465.89	201.64
	4458.11	4494.21
$p_c = 0.9, p_m = 0.05, P_{size} = 50$	4492.4	2300.22
	4458.11	4580.46
$p_c = 0.4, p_m = 0.15, P_{size} = 50$	4840.4	127874.85
	4458.11	5382.57
$p_c = 0.65, p_m = 0.15, P_{size} = 50$	4664.35	14633.46
	4460.92	4839.75
$p_c = 0.9, p_m = 0.15, P_{size} = 50$	4664.35	14633.46
	4460.92	4839.75
$p_c = 0.65, p_m = 0.05, P_{size} = 50$	4463.15	98.67
	4458.11	4481.27
$p_c = 0.65, p_m = 0.05, P_{size} = 100$	4568.57	10248.49
	4458.11	4762.35

best when the genetic probabilities are moderate and the population size is small. For other sets of GA parameters, the MGA demonstrates very consistent performance while the other two methods are highly affected by parameter changes. Therefore, the proposed MGA is more robust than existing methods under various experimental conditions and shows consistently outstanding results with different GA parameters.

4.2. Car diagnosis problem

The car diagnosis problem introduced by Norsys is considered in [15]. In this problem, the reason why a car does not move is presumed based on spark plugs, headlights, main fuse, among others. All nodes of the network are discrete variables. Some of them can take on three discrete states, while others can take on two states. A database of two thousand cases was utilized to train the BN. The database was generated using the Netica tool [15]. Fig. 5 shows the structure of the car diagnosis problem depicted by Netica from which sample cases were collected.

Fig. 6 shows the structure learning result of a specific run under the standard set of GA parameters (population size = 50, crossover rate = 0.65, mutation rate = 0.05). The short-dashed, long-dashed, and solid lines denote the performances of the methods in [7,10], and the MGA, respectively. As in Fig. 4, the vertical axis denotes the negative logarithm of the probability and a lower value reflects better performance.

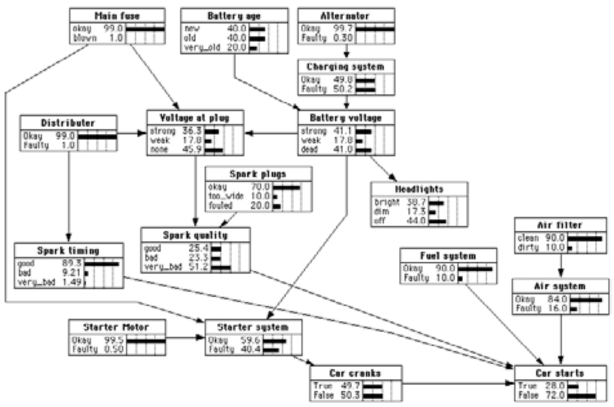

Fig. 5. The structure of the car diagnosis problem network.

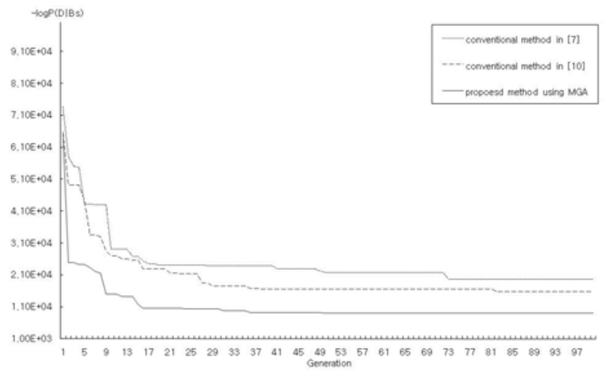

Fig. 6. Performance of the existing and proposed methods (car diagnosis problem).

The proposed method clearly outperforms the existing methods during evolution, as shown in Fig. 6. Table 5 compares the results using the methods in [7,10], and the proposed MGA in terms of (22). The proposed method

Table 5. The comparison of the proposed and existing methods (car diagnosis problem).

Trial time	Larrañaga *et al.* [7]	Lee *et al.* [10]	Proposed MGA
1	19687.91	15686.31	8954.81
2	18628.41	17093.18	8954.81
3	19864.35	17040.01	8954.81
4	18601.86	17311.46	8954.81
5	17214.43	15844.87	8954.81
6	19547.03	14520.13	8954.81
7	19462.29	19270.1	8954.81
8	20566.02	16871.59	8954.81
9	18916.31	16536.65	8954.81
10	16188.72	14707.35	8954.81
avg.	18867.73	16488.17	8954.81
var.	1711766.9	1926464.7	0
Best	16188.72	14520.13	8954.81
worst	20566.02	19270.1	8954.81

Table 6. The performances of the method in [7] with various parameters (car diagnosis problem).

GA parameters	Average	Variance
	Best	Worst
$p_c = 0.4, p_m = 0.05, P_{size} = 50$	18368.4	5304206.4
	14416.66	20408.46
$p_c = 0.9, p_m = 0.05, P_{size} = 50$	18869.21	1698531.13
	17563.87	21217.07
$p_c = 0.4, p_m = 0.15, P_{size} = 50$	31617.62	4114496.38
	29233.23	34629.94
$p_c = 0.65, p_m = 0.15, P_{size} = 50$	28866.97	3327884.4
	26714.66	32148.35
$p_c = 0.9, p_m = 0.15, P_{size} = 50$	32295.5	6573958.79
	29466.96	35996.97
$p_c = 0.65, p_m = 0.05, P_{size} = 50$	18867.73	1711766.9
	16188.72	20566.02
$p_c = 0.65, p_m = 0.05, P_{size} = 100$	19465.97	6056734.11
	15927.08	23306.73

Table 7. The performances of the method in [10] with various parameters (car diagnosis problem).

GA parameters	Average	Variance
	Best	Worst
$p_c = 0.4, p_m = 0.05, P_{size} = 50$	15917.75	5060879.36
	13464.59	19796.9
$p_c = 0.9, p_m = 0.05, P_{size} = 50$	18151.59	3865968.94
	14879.42	20149.28
$p_c = 0.4, p_m = 0.15, P_{size} = 50$	23942.66	2238232.94
	22402.89	26662.97
$p_c = 0.65, p_m = 0.15, P_{size} = 50$	25722.75	2898446.39
	22482.1	27508.1
$p_c = 0.9, p_m = 0.15, P_{size} = 50$	26672.84	9703812.46
	20857.47	29615.95
$p_c = 0.65, p_m = 0.05, P_{size} = 50$	16488.17	1926464.7
	14520.13	19270.1
$p_c = 0.65, p_m = 0.05, P_{size} = 100$	18606.82	3043408.32
	16049.6	21270.92

Table 8. The performances of the proposed MGA with various parameters (car diagnosis problem).

GA parameters	Average	Variance
	Best	Worst
$p_c = 0.4, p_m = 0.05, P_{size} = 50$	8954.81	0
	8954.81	8954.81
$p_c = 0.9, p_m = 0.05, P_{size} = 50$	8954.81	0
	8954.81	8954.81
$p_c = 0.4, p_m = 0.15, P_{size} = 50$	9417.52	53808.72
	9021.04	9698.79
$p_c = 0.65, p_m = 0.15, P_{size} = 50$	9417.52	53808.72
	9021.04	9698.79
$p_c = 0.9, p_m = 0.15, P_{size} = 50$	9417.52	53808.72
	9021.04	9698.79
$p_c = 0.65, p_m = 0.05, P_{size} = 50$	8954.81	0
	8954.81	8954.81
$p_c = 0.65, p_m = 0.05, P_{size} = 100$	8954.81	0
	8954.81	8954.81

outperforms the existing methods not only on average, but also in the best and worst cases.

Table 6, 7, and 8 show the learning results of the car diagnosis problem for a variety of GA parameters. The simulation parameters are same as the previous section.

Existing methods show the best performance when the genetic probabilities are low and the population size is small. For the proposed method, the effect of parameter variation is very small, and its performance is good and consistent compared to existing methods.

4.3. ALARM network

ALARM (A Logical Alarm Reduction Mechanism) is a medical diagnostic system for patient monitoring. It is a complex belief network with eight diagnoses, sixteen findings, and thirteen intermediate variables [21]. A database of two thousand cases was utilized to train the BN. As in the previous example, the database is generated using the Netica tool [15]. Fig. 7 shows the structure of the ALARM network depicted by Netica from which sample cases were collected.

Fig. 8 shows the learning result of the first run for the ALARM network problem. As in the previous examples, a lower vertical value means better performance as the negative logarithm of the probability is plotted. As before, in the case of the ALARM network, the proposed method shows good performance after tens of generations, as shown in Fig. 8. Table 9 compares the results from the methods in [7,10], and the proposed method in terms of (22). As seen in the previous two examples, the proposed method outperforms the existing methods not only on average, but also in the best and worst cases.

Table 10, 11, and 12 shows the learning results of the ALARM network problem for a variety of GA

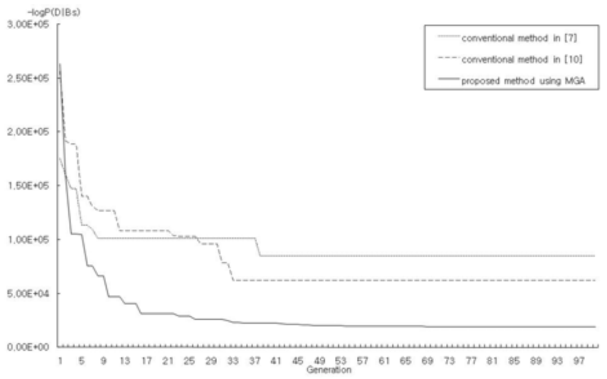

Fig. 8. Performance of the existing and proposed methods (ALARM network).

Table 9. The comparison of the proposed and existing methods (ALARM network).

Trial time	Larrañaga *et al.* [7]	Lee *et al.* [10]	Proposed MGA
1	84963.27	62540.11	18946.06
2	80986.9	79860.14	18946.06
3	82846.11	67474.26	18946.06
4	81709.27	75552.21	18946.06
5	84640.98	84972.31	18946.06
6	84479.74	74377.18	18946.06
7	79468.78	82558.87	18946.06
8	84059.88	70288.06	18946.06
9	86332.95	66382.3	18946.06
10	88639.38	68647.62	18964.73
avg.	83812.73	73265.31	18947.93
var.	7153099.99	55501237.33	34.86
Best	79468.78	62540.11	18946.06
worst	88639.38	84972.31	18964.73

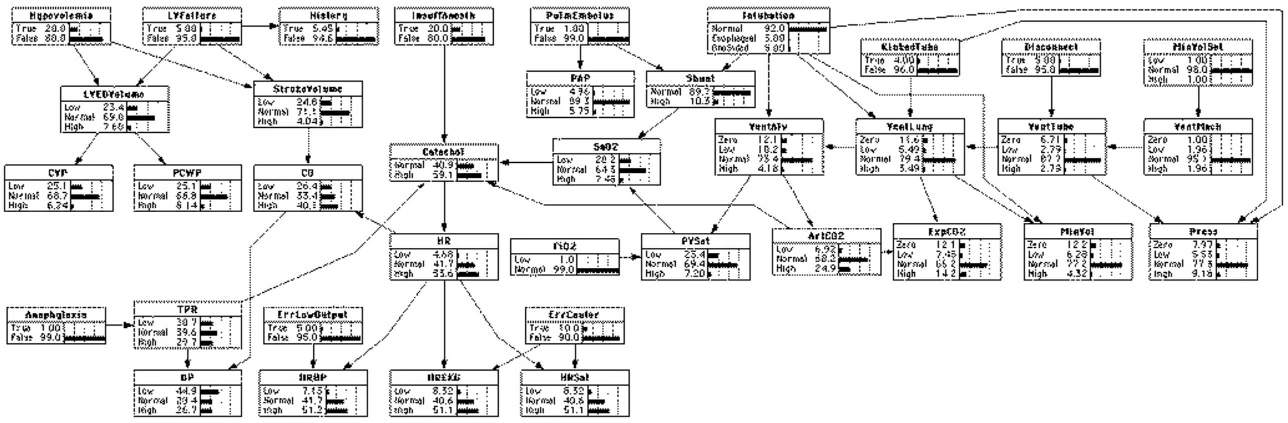

Fig. 7. The structure of the ALARM network.

parameters. Both the conventional and proposed methods show their best performances when the genetic probabilities are moderate and the population size is small. The proposed method shows good and consistent results compared with the other methods.

Table 10. The performances of the method in [7] with various parameters (ALARM network).

GA parameters	Average	Variance
	Best	Worst
$p_c = 0.4, p_m = 0.05, P_{size} = 50$	89026.93	9771180.3
	84063.54	93512.4
$p_c = 0.9, p_m = 0.05, P_{size} = 50$	88866.34	18727390.88
	84084.96	96007.34
$p_c = 0.4, p_m = 0.15, P_{size} = 50$	123635.32	46517874.85
	115573.5	135073.9
$p_c = 0.65, p_m = 0.15, P_{size} = 50$	125311.84	86297478.85
	115117.5	136623.2
$p_c = 0.9, p_m = 0.15, P_{size} = 50$	124341.48	61773623.57
	111196.1	132911.2
$p_c = 0.65, p_m = 0.05, P_{size} = 50$	83812.73	7153099.99
	79468.78	88639.38
$p_c = 0.65, p_m = 0.05, P_{size} = 100$	90171.19	12523301.27
	83567.76	96230.09

Table 11. The performances of the method in [10] with various parameters (ALARM network).

GA parameters	Average	Variance
	Best	Worst
$p_c = 0.4, p_m = 0.05, P_{size} = 50$	73120	55469600
	61100	82200
$p_c = 0.9, p_m = 0.05, P_{size} = 50$	75720	149209600
	59300	94300
$p_c = 0.4, p_m = 0.15, P_{size} = 50$	74220	143509600
	55400	87700
$p_c = 0.65, p_m = 0.15, P_{size} = 50$	83220	133897600
	63700	99300
$p_c = 0.9, p_m = 0.15, P_{size} = 50$	73620	51005600
	65900	82800
$p_c = 0.65, p_m = 0.05, P_{size} = 50$	73265.31	55501237.33
	62540.11	84972.31
$p_c = 0.65, p_m = 0.05, P_{size} = 100$	76239.48	138651798.4
	59301.32	91010.84

Table 12. The performances of the proposed MGA with various parameters (ALARM network).

GA parameters	Average	Variance
	Best	Worst
$p_c = 0.4, p_m = 0.05, P_{size} = 50$	18966.16	1123.12
	18946.06	19032.69
$p_c = 0.9, p_m = 0.05, P_{size} = 50$	18963.79	572.66
	18946.06	19010.65
$p_c = 0.4, p_m = 0.15, P_{size} = 50$	19674.71	668201.7
	18979.42	21180.93
$p_c = 0.65, p_m = 0.15, P_{size} = 50$	19674.71	668201.7
	18979.42	21180.93
$p_c = 0.9, p_m = 0.15, P_{size} = 50$	19599.53	124538.84
	19062.5	19998.55
$p_c = 0.65, p_m = 0.05, P_{size} = 50$	18947.93	34.86
	18946.06	18964.73
$p_c = 0.65, p_m = 0.05, P_{size} = 100$	12657.44	1039845.8
	10928.57	14303.32

5. CONCLUSIONS

In this paper, a new approach for structure learning of BN named was proposed, called MGA. In the proposed method, an individual is represented as a matrix chromosome and uses both the upper and lower triangular parts of the matrix. Further, new crossover and mutation operations were introduced to implement evolution of the matrix encoding. The MGA has an effective way of passing the good features of the parents to their offspring compared with previous methods, and thereby improves the performance of BN structure learning.

The proposed method was applied to three real world and benchmark problems. The simulation results demonstrated superior performance of the suggested method compared to previous methods for all three problems.

REFERENCES

[1] F. V. Jensen, "Introduction to Bayesian networks," *Technical Report IR 93-2003*, Dept. of Mathematics and Computer Science, Univ. of Aalborg, Denmark, 1993.

[2] M. L. Wong and K. S. Leung, "An efficient data mining method for learning Bayesian networks using an evolutionary algorithm-based hybrid approach," *IEEE Trans. on Evolutionary Computation*, vol. 8, pp. 378-404, Aug. 2004.

[3] H. Wang, D. Dash, and M. J. Druzdzel, "A method for evaluating elicitation schemes for probabilistic models," *IEEE Trans. on Systems, Man, and Cybernetics, Part B*, vol. 32, no. 1, pp. 38-43, Feb. 2002.

[4] S. Acid, L. M. De Campos, A. Gonzalez, R. Molina, and N. Perez de la Blanca, "Learning with CASTLE," *Symbolic and Quantitative Approaches to Uncertainty*, R. Kruse and P. Siegel, eds., Lecture Notes in Computer Science 548. Springer-Verlag, 1991.

[5] D. M. Chickering, D. Geiger, and D. Heckerman, "Learning Bayesian networks: search methods and experimental results," *Proc. Fifth Int'l Workshop Artificial Intelligence and Statistics*, pp. 112-128, 1995.

[6] G. M. Provan and M. Singh, "Learning Bayesian networks using feature selection," *Preliminary Papers Fifth Int'l Workshop Artificial Intelligence and Statistics*, pp. 450-456, 1995.

[7] P. Larrañaga, M. Poza, Y. Yurramendi, R. Murga, and C. Kuijpers, "Structure learning of Bayesian network by genetic algorithms: a performance analysis of control parameters," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 18, no. 9, pp. 912-926, Sept. 1996.

[8] R. Garza-Domínguez, M. Martínez-Morales, N. Cruz-Ramírez, J. L. Jiménez-Andrade, and A. G. Hernández, "A method based on genetic algorithms and fuzzy logic to induce Bayesian networks," *Proc. Fifth Mexican International Conference in Com-*

puter Science, Colima, Mexico, pp. 176-180, 2004.

[9] L. M. d. Campos, J. A. Gámez, and S. Moral, “Partial abductive inference in Bayesian belief networks - an evolutionary computation approach by using problem-specific genetic operators,” *IEEE Trans. on Evolutionary Computation*, vol. 6, pp. 105-131, April 2002.

[10] J. Lee, W. Chung, and E. Kim, “Structure learning of Bayesian networks using dual genetic algorithm,” *IEICE Trans. on Information and Systems*, vol. E91-D, no. 1, pp. 32-43, 2008.

[11] G. F. Cooper and E. A. Herskovits, “A bayesian method for the induction of probabilistic networks from data,” *Machine Learning*, vol. 9, no. 4, pp. 309-347, 1992.

[12] X. Li, X. He, and S. Yuan, “Learning Bayesian networks structures from incomplete data based on extending evolutionary programming,” *Proc. Int’l Conf. Machine Learning and Cybernetics*, vol. 4, pp. 2039-2043, Aug. 2005.

[13] S. Shetty and M. Song, “Structure learning of Bayesian networks using a semantic genetic algorithm-based approach,” *Proc. Int’l Conf. Information Technology: Research and Education*, pp. 454-458, 2005.

[14] W. Chung, *Context Aware Application for Smart Home Based on Bayesian Network*, Master thesis, Yonsei University, 2006.

[15] http://www.norsys.com/networklibrary.html

[16] F. V. Jensen, *Bayesian Networks and Decision Graphs*, Springer, 2001.

[17] S. Z. Zhang, Z. N. Zhang, N. H. Yang, J. Y. Zhang, and X. K. Wang, “An improved EM algorithm for Bayesian networks parameter learning,” *Machine Learning and Cybernetics*, vol. 3, pp. 1503-1508, Aug. 2004.

[18] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, Springer, 1999.

[19] P. Larrañaga, C. Kuijpers, and R. Murga, “Learning Bayesian network structures by searching for the best ordering with genetic algorithms,” *IEEE Trans. on Systems, Man, and Cybernetics, Part A*, vol. 26, pp. 487-493, 1996.

[20] P. Korpipää, M. Koskinen, J. Peltola, S. Mäkelä, and T. Seppänen, “Bayesian approach to sensor-based context awareness,” *Personal and Ubiquitous Computing*, vol. 7, no. 2, pp. 113-124, 2003.

[21] I. A. Beinlinch, H. J. Suermondt, R. M. Chavez, and G. F. Cooper, “The ALARM monitoring system: a case study with two probabilistic inference techniques for belief networks,” *Proc. Second European Conf. Artificial Intelligence in Medicine*, pp. 247-256, 1989.

[22] K. B. Hwang and B. T. Zhang, “Bayesian model averaging of Bayesian network classifiers over multiple node-orders application to sparse datasets,” *IEEE Trans. on Systems, Man, and Cybernetics, Part B*, vol. 35, no. 6, pp. 1302-1310, Dec. 2005.

Jaehun Lee received his B.S. and M.S. degrees in Electrical and Electronic Engineering from Yonsei University, Seoul, Korea, in 2005 and 2007, respectively. He is currently a Ph.D. candidate of the School of Electrical and Electronic Engineering at Yonsei University. His current research interests include computational intelligence, localization and tracking in wireless sensor network.

Wooyong Chung received his B.S. and M.S. degrees in Electrical and Electronic Engineering from Yonsei University, Seoul, Korea, in 2004 and 2006, respectively. He is currently a Ph.D. candidate of the School of Electrical and Electronic Engineering at Yonsei University. His current research interests include fuzzy control and evolutionary algorithm.

Euntai Kim received his B.S. (with top honors), M.S., and Ph.D. degrees in Electronic Engineering from Yonsei University, Seoul, Korea, in 1992, 1994, and 1999, respectively. From 1999 to 2002, he was a full-time lecturer with the Department of Control and Instrumentation Engineering at Hankyong National University, Gyeonggi-do, Korea. Since 2002, he has been with the School of Electrical and Electronic Engineering at Yonsei University, where he is currently an Associate Professor. He was a Visiting Scholar with the University of Alberta, Edmonton, Canada, and the Berkeley Initiative in Soft Computing (BISC), UC Berkeley, USA, in 2003 and 2008, respectively. His current research interests include computational intelligence and machine learning and their application to intelligent service robots, unmanned vehicles, home networks, biometrics, and evolvable hardware.

Soohan Kim received his B.S. degrees in Material Science from Chonnam National University, Gwangju, Korea, in 1990. He is currently a Senior Manager and Project Leader of the Internet Infra Technical Planning Team at Samsung Electronics Co. in Korea HQ. He joined Samsung in 1993 and was the first Software Development manager. His current research interests include Contents Sharing with 3 Screens, Multi-Media Platforms, localization and tracking in home network.