PBIL-RS: Effective Repeated Search in PBIL-based Bayesian Network Learning

Yuma Yamanaka[†*], Takatoshi Fujiki[†], Sho Fukuda[†], and Takuya Yoshihiro[‡**]

[†]Graduate School of Systems Engineering, Wakayama University, Japan
[‡]Faculty of Systems Engineering, Wakayama University, Japan
[*] s161065@sys.wakayama-u.ac.jp
[**] tac@sys.wakayama-u.ac.jp

Abstract - Bayesian Network, which is a model of probabilistic causal relationship, is an practically important graphical model learned from observation data. To learn a near-optimal Bayesian network model from a set of observation data, efficient optimization algorithm is required to search an exponentially large solution space, as this problem was proved to be NP-hard. To find better Bayesian network models in limited time, several efficient approximated search algorithms have been proposed such as genetic algorithms. Among them, algorithms based on probability vectors such as PBIL (Population-Based Incremental Learning) are regarded as a better sort of algorithms to learn superior Bayesian networks in a practical computation time. However, PBIL has a problem that it finishes executing when it converges. Namely, after convergence, it cannot find any better solutions. To solve the problem, in this paper, we propose PBIL-RS (PBIL-Repeated Search), which is an improvement of PBIL. In PBIL-RS, if the search area becomes sufficiently small in the process of converging the probability vector, we in turn spread the search area and again begin the converging process, repeatedly. We performed an evaluation of PBIL-RS and showed that it outperforms the existing algorithms in the PBIL-family. We further explored the behavior of PBIL-RS and found several key behaviors that lead the characteristics to find better solutions.

Keywords: Bayesian Networks, PBIL, Evolutionary Algorithm, EDA, Information Criterion

1 INTRODUCTION

Bayesian Network is used as a probabilistic model to analyze causal relationship between events from data. Recently, rapid growth of the Internet and processing speed of computers have made us possible to analyze the causal relationship from a large amount of data, and Bayesian Network is one of the important data analysis methods that are useful in various research fields with large data such as bioinformatics, medical analyses, document classifications, information searches, decision support, etc.

However, there is one difficulty that learning Bayesian Network models is proved to be NP-hard [1]. In other words, solution space exponentially increases as the number of events in the Bayesian Network increases. Therefore, several near-optimal algorithms to find better Bayesian Network models within a limited time have been proposed so far. Cooper et al. proposed an algorithm to learn Bayesian Networks called K2 that reduced execution time by limiting the search space [2]. To limit the search space, K2 applies a constraint in the order of events. The order constraint, for example, means that future events cannot be caused of events in the past. However, in many practical cases, we cannot assume such an order constraint. Therefore, to learn Bayesian Networks in general cases, several approaches have been proposed. Many of them use genetic algorithms (GAs), which find better Bayesian Network models when we take more time for computation [3]-[5]. Meanwhile, recently, requirements for large-data analyses arise due to the growth of the Internet. To meet these requirements, more efficient algorithms to find better Bayesian Network models within smaller time are strongly expected.

On the background above, a number of authors have proposed a new category of algorithms. Those algorithms called EDA (Estimation of Distribution Algorithm) have been reported to find better Bayesian Network models [6]-[8]. EDA is a kind of genetic algorithms that evolves statistic distributions from which we produce individuals over generations. Namely, EDA is a stochastic optimization algorithm. From the result of Kim et al., PBIL-based algorithm performed the best among several EDA-based algorithms [7].

Blanco et al. presented the first PBIL-based algorithm for Bayesian Networks [9]. They showed that their PBIL-based algorithm outperforms the traditional K2 algorithms. However, his algorithm has a drawback that his algorithm easily falls into local minimum solutions because it does not include any mutation operation to avoid converging into local minimum solutions. To overcome this drawback, several mutation operations were proposed for PBIL-based algorithms to learn Bayesian Networks. Handa et al. introduced bitwise mutation (BM), which apply mutation operations in which each edge is added or deleted with a constant probability [6]. Kim et al. proposed transpose mutation (TM) that is designed specific to Bayesian Networks [7]. This operation changes the direction of edges in the individuals produced in each generation. Fukuda et al. proposed a mutation operator called probability mutation (PM) for PBIL-based algorithms to learn Bayesian Networks [8]. Probability mutation manipulates the probability vector to avoid converging at local minimum solutions. These mutation operators improved the performance of PBIL-based algorithms to learn Bayesian Networks by avoiding local minimum solutions. However, mutation operators also have a drawback that the searching area jumps to other areas due to mutations before searching the local areas deep enough to explore good solutions.

In this paper, we propose a new PBIL-based algorithm called PBIL-RS (PBIL-Repeated Search), which is an improvement of PBIL. In PBIL-RS, if the search area becomes sufficiently small in the process of converging the probability vector, we in turn spread the search area and again begin the converging process, repeatedly. By searching local areas deeply until the probability vector converges into a sufficiently small area, PBIL-RS improves the performance of PBIL-based algorithms to learn Bayesian Networks. We performed an evaluation of PBIL-RS, clarified its characteristics, and showed the superiority in its performance.

The rest of this paper is organized as follows: In Section 2, we give the basic definitions on Bayesian Networks and also describe related work in this area of study. In Section 3, we propose a new efficient search algorithm called PBIL-RS to achieve better learning performance of Bayesian Networks. In Section 4, we describe the evaluation of PBIL-RS, and finally we conclude this paper in Section 5.

2 PRELIMINARY DEFINITIONS

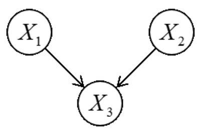
2.1 Bayesian Network

A Bayesian Network model visualizes the causal relationship among events through graph representation. In a Bayesian Network model, events are represented by nodes while causal relationships are represented by edges. See Fig. 1 for a concise example. Nodes X_1, X_2, and X_3 represent distinct events, where they take 1 if the corresponding events occur, and take 0 if the events do not occur. Edges $X_1 \to X_3$ and $X_2 \to X_3$ represent causal relationships, which mean that the probability of $X_3 = 1$ depends on events X_1 and X_2 . If edge $X_1 \to X_3$ exists, we call that X_1 is a parent of X_3 and X_3 is a child of X_1. Because nodes X_1 and X_2 do not have their parents, they have own prior probabilities $P(X_1)$ and $P(X_2)$. On the other hand, because node X_3 has two parents X_1 and X_2, it has a conditional probability $P(X_3|X_1, X_2)$. In this example, the probability that X_3 occurs is 0.890 under the assumption that both X_1 and X_2 occur. Note that, from this model, Bayesian inference is possible: if X_3 is known, then the posterior probability of X_1 and X_2 can be determined, which enables us to infer more accurately the occurrence of events.

The Bayesian Networks model can be learned from the data obtained through the observation of events. Let $O = \{o_j\}$, $(1 \leq j \leq S)$ be a set of observations, where S is the number of observations. Let $o_j = (x_{j1}, x_{j2}, \dots, x_{jN})$ be j-th observation, which is a set of observed values x_{ji} on event X_i for all $i(1 \leq i \leq N)$, where N is the number of events. We try to learn a good Bayesian Network model θ from the given set of observations. Note that, good Bayesian Network model θ is the one that creates data sets similar to the original observation O. As an evaluation criterion to measure the level of fitting between θ and O, we use AIC (Akaike's Information Criterion) [10], which is one of the best known criterion used in Bayesian Networks. Formally, the problem of learning Bayesian Networks that we consider in this paper is defined as follows:

Problem 1: From the given set of observations O, compute a Bayesian Network model θ that has the lowest AIC criterion value.

X_1	$P(X_1)$
0	0.79
1	0.21



X_2	$P(X_2)$
0	0.88
1	0.12

X_1	X_2	$P(X_3\|X_1, X_2)$	
		0	1
0	0	0.999	0.001
0	1	0.710	0.290
1	0	0.360	0.640
1	1	0.110	0.890

Figure 1: A Bayesian Network Model

2.2 PBIL

Recently, a category of the evolutionary algorithms called EDA (Estimation Distribution Algorithm) appears and reported to be efficient to learn Bayesian Network models. As one of EDAs, PBIL was proposed by Baluja et al. in 1994, which is based on genetic algorithm designed to evolve a probability vector [13]. Later, Blanco et al. applied PBIL to the Bayesian Network learning, and showed that PBIL efficiently works in this problem [9]. In PBIL, an individual creature s is defined as a vector $s = \{v_1, v_2, \dots, v_L\}$, where $v_i(1 \leq i \leq L)$ is the i-th element that takes a value 0 or 1, and L is the number of elements that consist of an individual. Let $P = \{p_1, p_2, \dots, p_L\}$ be a probability vector where $p_i(1 \leq i \leq L)$ represents the probability to be $v_i = 1$. The algorithm of PBIL is described as follows:

(1) As initialization, we let $p_i = 0.5$ for all $i = 1, 2, \dots, L$.

(2) Generate a set S that consists of C individuals according to probability vector P, i.e., element v_i of each individual is determined by the corresponding probability p_i.

(3) Compute the evaluation score for each individual $s \in S$ (In this paper we use AIC as the evaluation score).

(4) Select a set of individuals S' whose members have evaluation scores within top C' in S, and update the probability vector according to S' . Specifically, the formula applied to every p_i to update the probability vector is shown as follows.

$$p_i^{new} = ratio(i) \times \alpha + p_i \times (1 - \alpha), \qquad \text{(i)}$$

where p_i^{new} is the updated value of the new probability vector (p_i is soon replaced with p_i^{new}), $ratio(i)$ is the function that represents the ratio of individuals in S' that include edge i (i.e., $v_i = 1$), and α is the parameter called learning ratio.

(5) Repeat steps (2)-(4) until P converges.

By merging top-C' individuals, PBIL evolves the probability vector such that the good individuals are more likely to be generated. Different from other genetic algorithms, PBIL does not include "crossover" between individuals. Instead, it evolves the probability vector as a "parent" of the generated individuals.

2.3 PBIL-based Bayesian Network Learning

In this section, we describes a PBIL-based algorithm that learns Bayesian Network models. Because our problem (i.e. Problem 1) to learn Bayesian Network models is a little different from the general description of PBIL shown in the previous section, a little adjustment is required. In our problem, individual creatures correspond to each Bayesian Network model. Namely, with the number of events N, an individual model is represented as $s = \{v_{11}, v_{12}, \ldots, v_{1N}, v_{21}, v_{22}, \ldots, v_{N1}, v_{N2}, \ldots, v_{NN}\}$ where v_{ij} corresponds to the edge from an event X_i to X_j, i.e., if $v_{ij} = 1$, the edge from X_i to X_j exists in s, and if $v_{ij} = 0$ it does not exist. Similarly, we have the probability vector P to generate individual models as $P = \{p_{11}, p_{12}, \ldots, p_{1N}, p_{21}, p_{22}, \ldots, p_{N1}, p_{N2}, \ldots, p_{NN}\}$ where p_{ij} is the probability that the edge from X_i to X_j exists. A probability vector can be regarded as a table as illustrated in Fig. 2. Note that, because Bayesian Networks do not allow self-edges, p_{ij} is always 0 if $i = j$. The process of the proposed algorithm is basically obtained from the steps of PBIL, as described in the following.

(1) Initialize the probability vector P as $p_{ij} = 0$ if $i = j$, and $p_{ij} = 0.5$ otherwise, for each $i, j (1 \leq i, j \leq N)$.

(2) Generate S as a set of C individual models according to P. (This step (2) is illustrated in Fig. 3)

(3) Compute the evaluation scores for all individual models $s \in S$.

(4) Select a set of individuals S' whose members have top-C' evaluation values in S, and update the probability vector according to the formula (i). (These steps (3) and (4) are illustrated in Fig. 4.)

(5) Repeat steps (2)-(4) until P converges.

Same as PBIL, the proposed algorithm evolves the probability vector so that we can generate better individual models. However, there is a point specific to Bayesian Networks, that is, a Bayesian Network model is not allowed to have cycles in it. To consider this point in our algorithm, step 2 is detailed as follows:

(2a) Consider every pair of events (i, j) where $1 \leq i, j \leq N$ and $i \neq j$, create a random order of them.

(2b) For each pair (i, j) in the order created in step (2a), determine the value v_{ij} according to P; every time v_{ij} is determined, if v_{ij} is determined as 1, we check whether this edge from X_i to X_j creates a cycle with all the edges determined to exist so far. If it creates a cycle, let v_{ij} be 0.

(2c) Repeat steps (2a) and (2b) until all the pairs in the order are processed.

P		Parent Node					
		X_1	X_2	…	X_i	…	X_N
Child Node	X_1	0.0	0.5	…	p_{i1}	…	0.5
	X_2	0.5	0.0	…	p_{i2}	…	0.5
	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	…	$\vdots$
	X_j	p_{1j}	p_{2j}	…	p_{ij}	…	p_{Nj}
	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
	X_N	0.5	0.5	…	p_{iN}	…	0.0

Figure 2: A Probability Vector

These steps enable us to learn good Bayesian Network models within the framework of PBIL. Note that the algorithm introduced in this section does not include mutation operators. Therefore, naturally, it easily converges to a local minimum solution. To avoid converging to the local minimum solution and to improve the performance of the algorithm, several mutation operations have been proposed. A mutation operator called bitwise mutation (BM) was introduced by Handa [6]. BM applies mutations to each edge in each individual with a certain mutation probability. Kim et al. proposed a mutation operator called transpose mutation (TM), which is specifically designed for Bayesian Networks [7]. TM changes the direction of edges in the individuals produced in each generation. Fukuda et al. proposed a mutation operator called probability mutation (PM) for PBIL-based Bayesian Network learning [8]. PM manipulates the probability vector to avoid converging at local minimum solutions. These mutations avoid converging at local minimum solutions, and it improves the efficiency to learn Bayesian Networks with PBIL-based algorithms.

3 PROPOSED ALGORITHM: PBIL-RS

We propose PBIL-RS (PBIL- Repeated Search), which is an algorithm to learn Bayesian Networks based on PBIL. To search for good Bayesian Networks efficiently, we introduce a new technique instead of mutation operators. Because mutation operators work with a certain mutation probability, they tend to change the search space before we deeply search the current search area to explore good solutions. As a result, efficiency of the algorithm decreases by skipping the search areas where many superior solutions are likely to be buried. In contrast, in PBIL-RS, we transit the search space only after we search the current search area deeply, i.e., only after PBIL-RS judged that the search space gets converged. With this technique, we can search deeply the specific space in which superior solutions would exist while avoiding local minimums.

Figure 5 shows the outline of PBIL-RS. In general, in the search space of Bayesian Network models, there are many local minimum points. Because models with similar structures tend to have similar evaluation scores, superior solutions would likely be collected at several local areas in the search

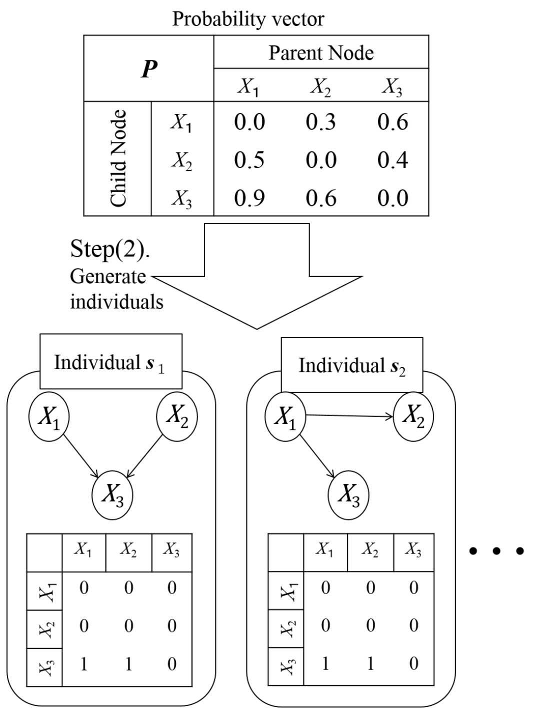


Figure 3: Generating Individuals from Probability Vector

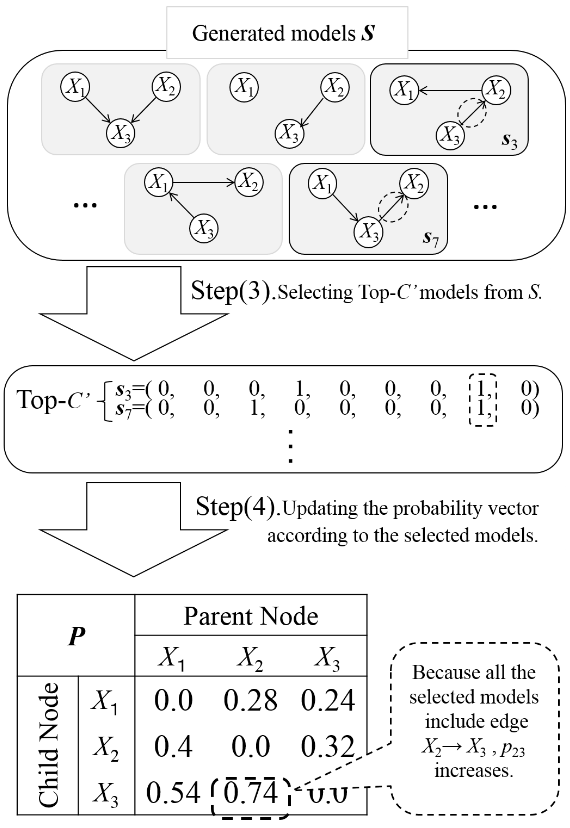

Generated models S
s_3
s_7
Step(3).Selecting Top-C' models from S.
Top-C' s_3=(0, 0, 0, 1, 0, 0, 0, 1, 0)
s_7=(0, 0, 1, 0, 0, 0, 0, 1, 0)
Step(4).Updating the probability vector according to the selected models.
P
Parent Node
Child Node
X_1 X_2 X_3
0.0 0.28 0.24
0.4 0.0 0.32
0.54 0.74 0.0
Because all the selected models include edge $X_2 \rightarrow X_3$, p_{23} increases.

Figure 4: Step(3)(4): Updating Probability Vector

space. Our algorithm PBIL-RS explores these areas with the following steps: (1) Initially, PBIL-RS sets the search space as the whole solution space. (2) As the algorithm proceeds and the generation grows, the search space usually gets smaller by focusing on an area in which superior solutions would be likely to exist. When the search space converges to a sufficiently small area, and PBIL-RS judges that the current area is sufficiently searched out, and (3) PBIL-RS in turn spreads the search space to explore different local minimum areas. Here, if the size of the spread search space is not sufficiently large, it may again fall into the same local minimum area. In order to avoid this, PBIL-RS spreads it to be larger search spaces step-by-step. Specifically, the size of the spread search space is firstly small to search near local minimum areas, and if we cannot find superior solutions in the next convergence, we then try to spread to larger search spaces to reach more distant search areas.

PBIL-RS controls the search space with probability vector P. Each element $p_{i,j}$ of vector P represents the probability to have the corresponding edge (i, j) in the generated Bayesian Network models. Thus, if each element $p_{i,j}$ approaches to 0 or 1, then naturally we have a probabilistic bias in the structure of the generated Bayesian Network models: The closer to 0.5 each element of probability vector P is, the larger the variation of generated models, and the closer to 0 or 1 each element is, the smaller the variation is. Namely, the probability vector P controls the variation and the bias of the generated structures of Bayesian Network models. Based on this, for probability vector P, we define *convergence level* S as follows:

$$S = \frac{\Sigma_{i,j(i \neq j)}\{0.5 - |P_{ij} - 0.5|\}}{N(N-1)}. \qquad \text{(ii)}$$

Convergence level S takes the average of the difference between 0 (or 1) and each element of probability vector P. Namely, the less this value is, the smaller search space is. In PBIL-RS, generally *convergence level* S gets smaller as generation proceeds. Thus, PBIL-RS spreads the search space when the search space shrinks to be sufficiently small. To judge that the search space is sufficiently small, we introduce the number of search limitation k. Specifically, when *convergence level* S does not update the smallest value in the past k generations, i.e., the *convergence level* S in the k-th last generation takes the smallest value in the past k generations, PBIL-RS judges that the search space is sufficiently small and has converged.

When PBIL-RS detects the search space convergence, it in turn spreads the search space. We define H as the level to spread the search space. PBIL-RS modifies the probability vector P to increase the *convergence level* S to H. Specifically, we choose an element of P randomly, and reset it as $P_{ij} = 0.5$. This operation repeats until $S \leq H$ holds.

In addition, as mentioned previously, we change the value H dynamically to spread the search space and so avoid converging to the same local minimum areas repeatedly. More specifically, (a) we firstly initialize H with the initial value H_{min}, (b) secondly every time the search space is converged we increase H by a constant value *spread width* H_{inc}, and (c) lastly when we find the solution that has the best score so far, we again initialize H with the initial value. This operation enables PBIL-RS to leave a local area quickly when good solutions would hardly be found, and guide to the bigger search

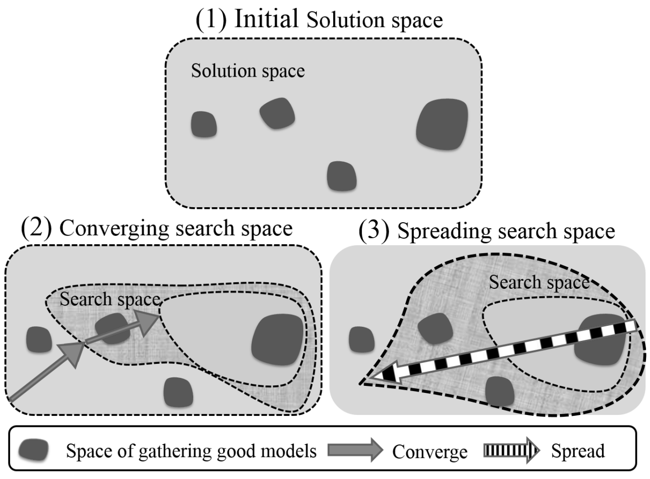

(1) Initial Solution space
Solution space
(2) Converging search space
Search space
(3) Spreading search space
Search space
Space of gathering good models
Converge
Spread

Figure 5: PBIL-RS Method

space.

The formal description of PBIL-RS is as follows. Processes (i)-(iv) are inserted into the steps (4) and (5) described in subsection 2.3.

(i) If the Bayesian Network model that has the best score so far is found, H is initialized by the initial value H_{min}.

(ii) Choose an element p_{ij} in P randomly, and reset it as $p_{ij} = 0.5$.

(iii) If $S < H$, then return to step (ii).

(iv) Augment H by *spread width* H_{inc}.

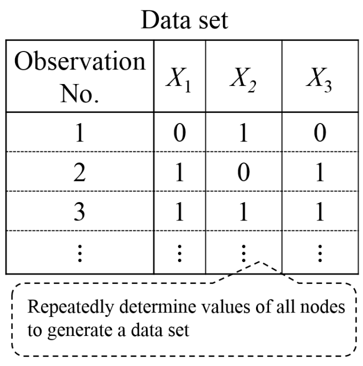
4 EVALUATION

4.1 Purpose of Evaluations

We evaluate PBIL-RS to measure the performance and also to explore the key behavior of PBIL-RS to find better solutions. In our evaluation, we first clarify that PBIL-RS performs well even if the number of events vary, and next we investigate the effect of several parameter variables on the performance. Clarifying the good parameter values would be helpful for us in determining the parameter values in the practical scenes. Through these evaluation results, we would find that the behavior of PBIL-RS is favorable to find better Bayesian Network models efficiently.

4.2 Performance Comparison with Existing Methods

We designed our evaluation procedure as follows: We select Bayesian Network models used in our evaluation. In this paper, we use two well-known Bayesian Network models called Alarm Network [11] and Pathfinder [12], where Alarm Network represents the causal relation among events to monitor patients in intensive care units, and Pathfinder represents that related to the diagnosis of lymph node diseases. Note that Alarm Network includes 37 nodes and Pathfinder does



Data set

Observation No.	X_1	X_2	X_3
1	0	1	0
2	1	0	1
3	1	1	1
⋮	⋮	⋮	⋮

Repeatedly determine values of all nodes to generate a data set

Figure 6: Generating an Observation Data Set

Table 1: Parameters of PBIL-RS

Parameters	Values
# of observations	1000
Individuals in a generation (C)	1000
# of selected individuals (C')	10
Learning Ratio (α)	0.1
Search limitation (k)	10
Initial spreading level (H_{min})	0.2
Spread width (H_{inc})	0.05
Evaluation Score	AIC

135 nodes. We generate an observation data set from each of the two models. In a Bayesian Network model, each node has a set of conditional probability so that we can obtain a set of values corresponding to all nodes according to the probability. Figure 6 shows an example of the data set generated from the example of Bayesian Networks shown in Fig 1, where j-th row represents an example of j-th observation set o_j generated according to the conditional probabilities of the model. We generated a data set that consists of 1,000 observations from each of two models. We use AIC criterion as the evaluation score, which is one of the representative criterion to measure the distance between the input data set and a Bayesian Network model.

We perform an evaluation of PBIL-RS in comparison with existing methods. We compare the performance of PBIL-RS with K2 that order restriction is evolved by genetic algorithms (K2-GA) [3], PBIL without mutations, and PBIL with three different mutation operators BM, TM, and PM. Parameter values used in the evaluation are shown in Table 1. Note that the mutation probability for BM, TM, and PM that performs the best is different for each mutation operators. Thus, we carefully chose those through preliminary experiments. For BM we use 0.005 that is the best performance mutation probability in range [0.001:0.2]. Similarly, for TM and PM, we use 0.1 and 0.002 that are the best in range [0.001:0.2] and [0.001:0.009], respectively.

Table 2 shows the comparison result summarizing the value of AIC calculated by each method. In Table 2, we show the performance of each method running 500 generations for two Bayesian Network models. In this result, we use the mean

Table 2: AIC Values at 500 Generations

Methods	Bayesian Network Models	
	Alarm (37 events)	Pathfinder (135 events)
PBIL-RS	8536.4	30138.6
PBIL	8627.2	30243.7
PBIL + BM	8563.1	35240.9
PBIL + TM	8654.3	34784.2
PBIL + PM	8582.9	33003.0
K2-GA	13347.7	-

Table 3: Computation Time (sec) for 500 Generations

Methods	Bayesian Network Models	
	Alarm (37 events)	Pathfinder (135 events)
PBIL-RS	8914.9	125195.6
PBIL	2791.9	75618.1
PBIL + BM	7992.4	156114.0
PBIL + TM	7920.7	161241.2
PBIL + PM	8724.9	142665.1
K2-GA	474327.0	-

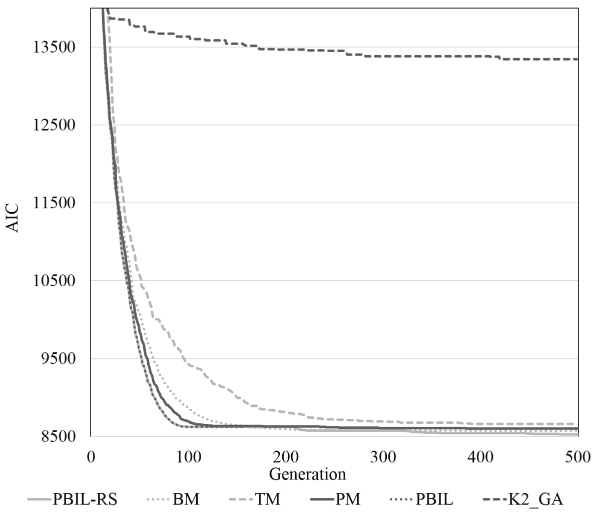


Figure 7: AIC Transition in Case of Alarm Network

Table 4: Variances of AIC Values at 400 Generations

Methods	Bayesian Network Models	
	Alarm (37 events)	Pathfinder (135 events)
PBIL-RS	508.3	43594.1
PBIL + BM	2708.1	56006.6
PBIL + TM	7458.2	235816.8
PBIL + PM	2498.9	108321.9

of 10 repetitions. Also, in Fig. 7, we show the transition of AIC values in the case of Alarm Network. From these results, we found that the PBIL-series methods perform far better than the traditional K2 although its order restriction is evolved by genetic algorithms, which proves the excellent ability of PBIL-based algorithms. Note that we could not compute the score of K2-GA for Pathfinder because it requires very large amount of time; it took 650 hours to proceed only 45 generations, whereas PBIL-RS took only 3 hours.

We also found that PBIL-RS has the best performance among those PBIL-based algorithms. This is because PBIL cannot continue searching after convergence (e.g., it finishes running at 160 generations in Alarm and 302 generations in Pathfinder), while BM, TM, and PM frequently change the searching area before exploring there deeply.

In Table 3, we show the execution time of those algorithms for 500 generations. PBIL finished in especially short time, which is because it finishes execution whenever the search space converges. On the other hand, K2-GA takes very long time because of large searching time of K2 algorithm. Except for those two, we found that the execution time of the variations of PBIL are comparable.

Table 4 shows the variance of AIC scores at the 400th generation with 10 repeated executions. From this result, we found that the variance of PBIL-RS is the smallest in both Alarm and Pathfinder, meaning that PBIL-RS most stably computes good solutions.

As above, we showed that PBIL-RS has the best performance among the methods compared in this evaluation. In the following sections, we examine the effect of several essential parameters on the performance of PBIL-RS, and investigate the key behavior of PBIL-RS that contributes to the superior ability.

4.3 Effect of Learning Ratio α

We examine the effect of two essential parameters through several evaluations. First, we focus on the effect of Learning Ratio α. We execute PBIL-RS with several learning ratios in range [0.05:0.7], and compare AIC values of Bayesian Network models.

Figure 8 shows the transition of AIC scores of each Bayesian Network models as generation proceeds. In Fig. 8, we show the average AIC values of 30 repetitions, where the horizontal axis represents generations, and the vertical axis represents the best AIC score found as generation proceeds. Also, in Fig. 9, we show the AIC scores at 500th generation for each learning ratios.

From those results, we found the property that the final AIC scores are better when learning ratio α takes lower values. Simultaneously, however, if α takes lower values, the speed to find better solutions goes slower than the case of higher α. From this trade-off, we found that lower learning ratio α is better, but if we have limitation on the executable generations, we have to determine α carefully.

4.4 Effect of Spreading Level H

PBIL-RS increases the spreading level step-by-step to search larger search spaces to avoid falling into the same local mini-

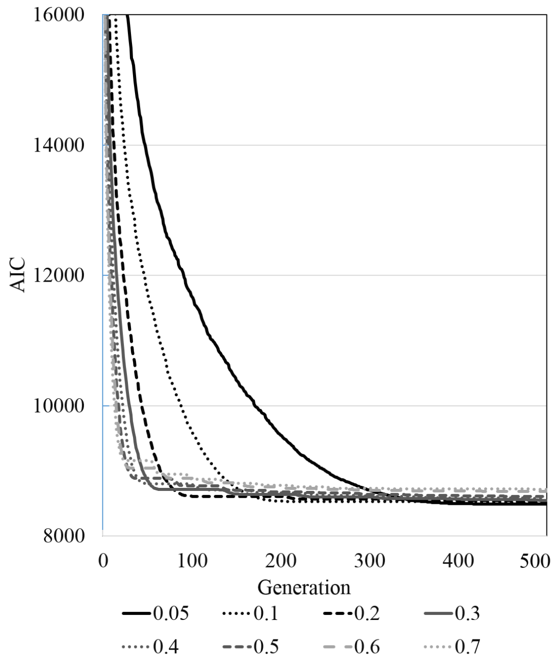


Figure 8: AIC Values under Variation of Learning Ratio α

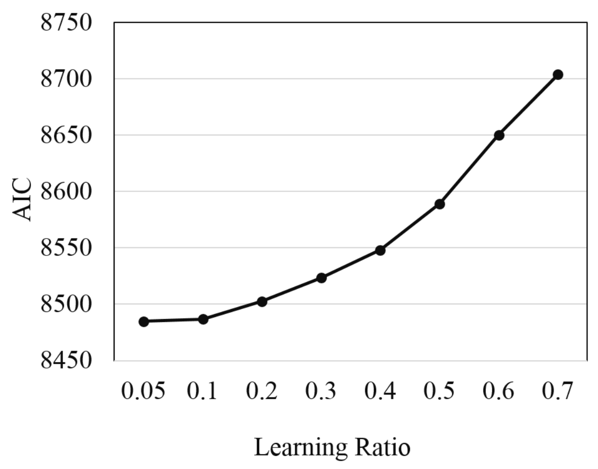


Figure 9: AIC Values with Learning Ratio at 500 Generation

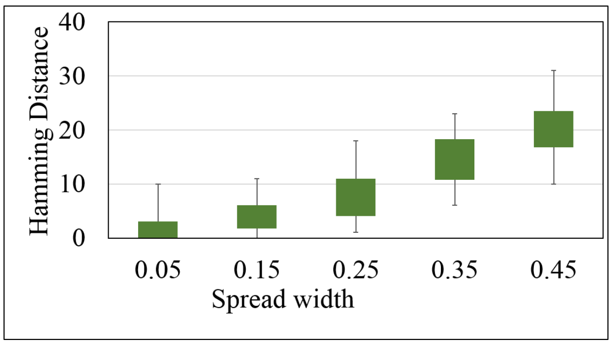


Figure 10: Hamming Distances among Convergence Points under Variation of Spreading Levels

mum area. In this section, we clarify the relationship between spreading level and the similarity of the Bayesian Network model structures in PBIL-RS. Also, by focusing on the convergence point of probability vector P, we find the key property of PBIL that show the superior performance of PBIL-RS.

Recall that each element p_{ij} of the probability vector gets closer to 0 or 1 as the algorithm proceeds and generation grows. In most cases, p_{ij} actually converges to 0 or 1, where P always generate the same individual. Thus, we can regard each convergence point as a string of binary digits. We measure the similarity of two convergence points using Hamming distance of the corresponding strings in order to grasp the distribution of the convergence points under various values of the spreading level.

We examine the distance among convergence points under variation of spreading levels. Specifically, we varied the *initial spreading level* H_{min} in [0.05:0.45] under fixed *spread width* $H_{inc} = 0$, and ran PBIL-RS with 1000 generations to examine the Hamming distance between every pair of the convergence points in a single run. Figure 10 shows the result as a box plot where the above and the below of the boxes show the maximum and minimum values, and the top and the bottom of the boxes show the values of the first and third quartiles of all the data. From Fig. 10, we see that the spreading level is clearly related to the Hamming distance between the convergence points, meaning that large spreading levels has an ability to change search space and to avoid converging to local minimum areas.

Next, we examine the behavior of the algorithm, especially on the timing at which good models are found. We ran PBIL-RS for 1000 generations with parameter values $H_{min} = 0.2$ and $H_{inc} = 0$. Figure 11 shows the result where the line represents the transition of *convergence level* S and the dotted points represents the timing where the best model is updated. From Fig. 11, we see the basic behavior of PBIL-RS such that each time P converges it spreads the search space. Please pay attention to the behavior of PBIL-RS that the point it updates the best models is concentrated on where the *convergence level* S is very small. Figure 12 is the scatter diagram that shows the distribution of the updated points. Each dotted point represents the updated point, and the horizontal axis shows the the updated amount of AIC, i.e., the difference between the new AIC value and the previous one. From Fig. 12, one sees that the best Bayesian Network model is updated when *convergence level* S is in the range of small values [0.0005:0.015]. The above results have clarified that deep exploration of local areas is an efficient strategy to find good Bayesian Network models.

4.5 Behavior in Varied Spread Level

In the previous section, we found two important properties of PBIL-RS that, first, large spread levels have an ability to avoid local minimum areas, and second, deep exploration of a local area is a preferable strategy to find good models, which support the superior performance of PBIL-RS.

In this section, we examine the behavior of the original PBIL-RS in which the value of spreading width is varied. We ran PBIL-RS in 3000 generations. Figure 13 shows the result

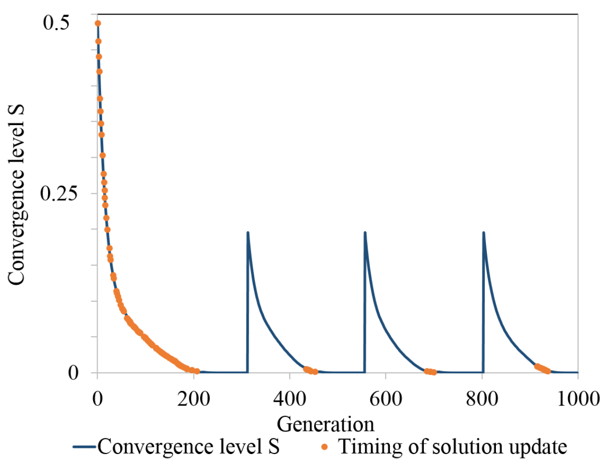


Figure 11: Transition of Convergence Levels, and the Updated Timing

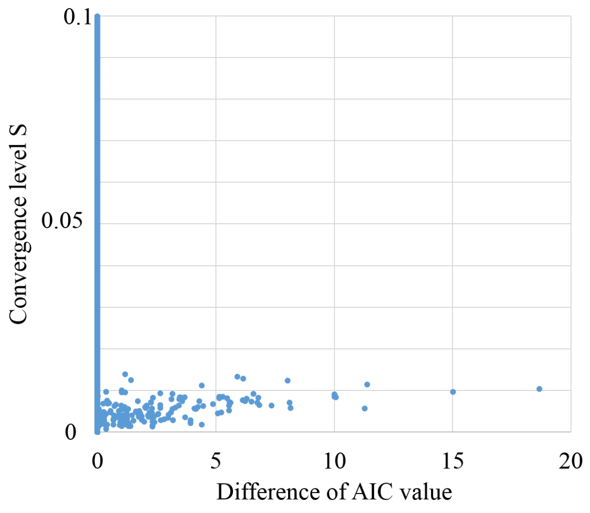


Figure 12: Convergence Levels at Each Update Point

where the line shows the transition of convergence levels as generation proceeds, and the dotted points shows the timing at which the best model is updated. From Fig. 13, we see that, as we found in the previous section, PBIL-RS finds the best model when the spread level comes to be low, and PBIL-RS continues finding better models even around 3000th generations by changing the exploring areas adaptively. We conclude that the good properties found in the previous section also work in the original behavior of PBIL-RS in which the convergence level is changed adaptively.

5 CONCLUSION

In this paper, we proposed a new algorithm called PBIL-RS, which is an algorithm to learn Bayesian Network models. PBIL-RS is an extension of PBIL that avoids convergence to local minimum solutions by means of spreading the search space repeatedly whenever it converges to a small area. Note that PBIL-RS is somewhat similar to the simulated annealing, but PBIL-RS is different from it in that PBIL-RS uses statistic property to find good solutions whereas the simulated annealing uses only the random effect. We evaluated the performance of PBIL-RS in comparison with existing algorithms, and we showed that PBIL-RS outperforms other existing algorithms regardless of the number of nodes in the Bayesian Network models used in the evaluation. In addition, we showed that the learning ratio significantly effects on efficiency of the algorithm, which clarified that selecting suitable learning ratio according to the planned execution time is important. Moreover, by examining the behavior of PBIL-RS, we verified that it properly controls the search space depending on the situation, and which leads to the superior performance.

As future work, more extensive evaluation using various Bayesian Network models is important. Especially, we would like to apply the models that include several thousands of nodes.

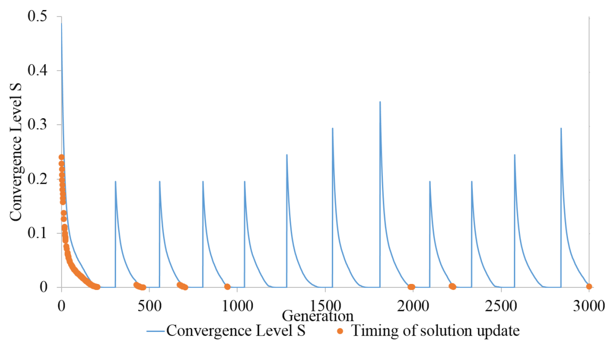


Figure 13: Behavior of PBIL-RS in Changing Convergence Levels

ACKNOWLEDGEMENT

This work was partly supported by “the Program for Promotion of Stockbreeding” of JRA (Japan Racing Association).

REFERENCES

[1] D.M. Chickering, D. Heckerman, C. Meek, “Large-Sample Learning of Bayesian Networks is NP-Hard,” Journal of Machine Learning Research, Vol.5, pp.1287-1330 (2004).
[2] G.F. Cooper, and E. Herskovits, “A Bayesian Method for the Induction of Probabilistic Networks from Data,” Machine Learning, Vol.9, pp.309-347 (1992).
[3] W.H. Hsu, H. Guo, B.B. Perry, and J.A. Stilson, “A Permutation Genetic Algorithm for Variable Ordering in Learning Bayesian Networks from Data,” In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO’02), pp.383-390 (2002).
[4] O. Barriére, E. Lutton, P.H. Wuillemin, “Bayesian Network Structure Learning Using Cooperative Coevolution,” In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO’09), pp.755-762 (2009).

[5] A.P. Tonda, E. Lutton, R. Reuillon, G. Squillero, and P.H. Wuillemin, "Bayesian Network Structure Learning from Limited Datasets through Graph Evolution," In Proceedings of the 15th European conference on Genetic Programming (EuroGP '12), pp.254-265 (2012).
[6] H. Handa, "Estimation of Distribution Algorithms with Mutation," Lecture Notes in Computer Science, Vol.3448, pp.112-121 (2005).
[7] D.W. Kim, S. Ko, and B.Y. Kang, "Structure Learning of Bayesian Networks by Estimation of Distribution Algorithms with Transpose Mutation," Journal of Applied Research and Technology, Vol.11, pp.586-596 (2013).
[8] S. Fukuda, Y. Yamanaka, and T. Yoshihiro, "A Probability-based Evolutionary Algorithm with Mutations to Learn Bayesian Networks," International Journal of Artificial Intelligence and Interactive Multimedia, Vol.3, No.1, pp.7-13 (2014).
[9] R. Blanco, I. Inza, P. Larrañaga, "Learning Bayesian Networks in the Space of Structures by Estimation of Distribution Algorithms," International Journal of Intelligent Systems, Vol.18, pp.205-220 (2003).
[10] H. Akaike, "Information theory and an extension of the maximum likelihood principle," Proceedings of the 2nd International Symposium on Information Theory, pp.267-281 (1973).
[11] I.A. Beinlich, H.J. Suermondt, R.M. Chavez, G.F. Cooper, "The ALARM Monitoring System: A Case Study with Two Probabilistic Inference Techniques for Belief Networks," In Proc. of Second European Conference on Artificial Intelligence in Medicine, Vol. 38, pp.247-256 (1989).
[12] D.E. Heckerman, E.J. Horvitz, B.N. Nathwani, "Towards Normative Expert Systems: Part I - the Pathfinder Project," Methods of Information in Medicine, pp. 90-105 (1992).
[13] S. Baluja, "Population-Based Incremental Learning: A method for Integrating Genetic Search Based Function Optimization and Competitive Learning," Technical Report CMU-CS-94-163, Carnegie Mellon University (1994).

(Received September 22, 2015)

Yuma Yamanaka received the B.E. degree from Wakayama University in 2015. He is currently a Master-course student in Wakayama University.

Takatoshi Fujiki recieved his B.E. and M.E. degrees from Wakayama University in 2010 and 2012, respectively. He is currently a doctoral course student in Wakayama University. He is interested in data mining, machine learning, and bioinformatics. He is a student member of IPSJ.

Sho Fukuda received his B.E. and M.E. degrees from Wakayama University in 2012 and 2014, respectively. He is currently working with Intec Hankyu Hanshin Co.Ltd. He is interested in Data Analytics with large data sets.

Takuya Yoshihiro received his B.E., M.I. and Ph.D. degrees from Kyoto University in 1998, 2000 and 2003, respectively. He was an assistant professor in Wakayama University from 2003 to 2009. He has been an associate professor in Wakayama University from 2009. He is currently interested in the graph theory, dis- tributed algorithms, computer networks, medial applications, and bioinformatics, and so on. He is a member of IEEE, IEICE, and IPSJ.