Yang JY, Wang JD, Zhang YF *et al.* A heuristic sampling method for maintaining the probability distribution. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 36(4): 896–909 July 2021. DOI 10.1007/s11390-020-0065-6

A Heuristic Sampling Method for Maintaining the Probability Distribution

Jiao-Yun Yang[1,2,3], *Member, CCF*, Jun-Da Wang[1,2,4,*], Yi-Fang Zhang[1,2,3], Wen-Juan Cheng[1,2,3], and Lian Li[1,2,3], *Member, CCF*

[1] *Key Laboratory of Knowledge Engineering with Big Data of Ministry of Education, Hefei University of Technology Hefei 230601, China*

[2] *National Smart Eldercare International Science and Technology Cooperation Base, Hefei University of Technology Hefei 230601, China*

[3] *School of Computer Science and Information Engineering, Hefei University of Technology, Hefei 230601, China*

[4] *School of Mathematics, Hefei University of Technology, Hefei 230601, China*

E-mail: jiaoyun@hfut.edu.cn; jwang212@ur.rochester.edu; u2034168@live.warwick.ac.uk; cheng@ah.edu.cn llian@hfut.edu.cn

Received October 5, 2019; accepted August 15, 2020.

Abstract Sampling is a fundamental method for generating data subsets. As many data analysis methods are developed based on probability distributions, maintaining distributions when sampling can help to ensure good data analysis performance. However, sampling a minimum subset while maintaining probability distributions is still a problem. In this paper, we decompose a joint probability distribution into a product of conditional probabilities based on Bayesian networks and use the chi-square test to formulate a sampling problem that requires that the sampled subset pass the distribution test to ensure the distribution. Furthermore, a heuristic sampling algorithm is proposed to generate the required subset by designing two scoring functions: one based on the chi-square test and the other based on likelihood functions. Experiments on four types of datasets with a size of 60 000 show that when the significant difference level, α, is set to 0.05, the algorithm can exclude 99.9%, 99.0%, 93.1% and 96.7% of the samples based on their Bayesian networks—ASIA, ALARM, HEPAR2, and ANDES, respectively. When subsets of the same size are sampled, the subset generated by our algorithm passes all the distribution tests and the average distribution difference is approximately 0.03; by contrast, the subsets generated by random sampling pass only 83.8% of the tests, and the average distribution difference is approximately 0.24.

Keywords Bayesian network, chi-square test, sampling, probability distribution

1 Introduction

Sampling is a fundamental method in data science, especially in the big data era. Sampling generates small subsets that are intended to represent the original whole datasets to reduce computational complexity. Sampling has been widely applied to many applications. An important application is data trading or data exchange [1]. In a data center, data sellers or suppliers are often required to provide a subset of a dataset to show the data characteristics. As many data analysis methods are developed based on probability distributions, an intuition behind this is that if the distributions of the subset and the original whole dataset are consistent or similar, the characteristics of the subset may represent most of the characteristics of the whole dataset. Providing the subset rather than the whole dataset may even be sufficient for data sellers or suppliers to meet the data buyers' or demanders' analysis requirements. In this paper, we focus on the sampling methods that

Regular Paper

Recommended by NCTCS 2019

This work was supported by the National Key Research and Development Program of China under Grant No. 2018YFB1003204, Anhui Provincial Key Technologies Research and Development Program under Grant Nos. 1804b06020378 and 1704e1002221, and the National 111 Project of China under Grant No. B14025.

[*]Corresponding Author

can ensure the distributions.

Based on statistical properties, the sampling methods can be divided into two categories: probability sampling and nonprobability sampling [2]. The difference between the two categories is whether some samples have no chance to be selected. The most typical probability sampling is random sampling in which each sample has an equal probability to be chosen. However, random sampling's performance is unstable. Various sampling methods are designed to improve sampling quality. Systematic sampling involves sorting samples according to a set of rules and then choosing samples at regular intervals [3]. Stratified sampling involves dividing samples into various categories or strata and applying random sampling in each category at a specific sampling ratio [4]. This sample partitioning should minimize the varieties within categories and maximize varieties between categories. Clustering methods may be adopted for this partition task [5, 6]. However, stratified sampling is different from cluster sampling, which involves choosing a whole cluster at a time. Generally, these probability sampling methods all involve random sampling. The larger the size of a chosen subset is, the closer the distribution is to the original distribution. However, these methods have no mechanisms for minimizing the subpopulations' size while maintaining the probability distribution.

Nonprobability sampling methods are often designed according to the purpose of an application. Snowball sampling, which is usually applied to recruit subjects [7, 8], is a widely-used nonprobability sampling method in the social sciences. In snowball sampling, an initial subject group is first determined, and then more subjects are recruited based on the previously chosen subjects. Sampling is also needed in some data science applications to choose typical samples to improve the performance of machine learning models, e.g., active learning models. Sampling methods developed for these applications are usually designed based on special rules [9–11], e.g., choosing samples close to the decision boundary or choosing samples belonging to the center in different clusters. These methods are not designed for maintaining the distribution, and they cannot be generalized for general applications. In addition, some researchers try to determine the minimum sampling size based on a fixed standard of data analysis performance [12–14]. Silva *et al.* used the performance of machine learning methods as the criterion for choosing samples by using a heuristic search approach [12]. Alwosheel *et al.* determined the minimum size for artificial neural networks by using Monte Carlo experiments [13]. The findings of these studies are based on the use of specific machine learning models. When the model is changed, the sampled subsets may become unsatisfactory based on the new model's criteria.

Probability sampling methods are more likely to maintain the probability distribution; however, the performance is unstable, and these types of methods usually need to include many samples. Recently, Yang *et al.* proposed to use the chi-square test to guide the sampling process [15]. Their method is a greedy method. They randomly sampled some subsets in each iteration and added the best one into the final subset. As the subsets are randomly sampled, some samples may not be good for the final subset to pass the distribution test. Although they eliminated some samples with worse scoring values in each iteration, their method could not still avoid these samples.

Some researchers also proposed methods that, unlike those that extract samples from the whole population, generate samples according to the probability distribution, including Markov chain Monte Carlo (MCMC) sampling [16–18] and Gibbs sampling [19, 20]. These methods can ensure that the distribution of the generated sample set is the same as the original distribution. When generating new samples, these methods compute the posterior probabilities based on current samples to obtain a new sample. A problem is that these generated samples may not exist in the original dataset, as they are generated by probabilities.

In this paper, we aim to develop a sampling method that ensures that the distribution of the sampled subset is the same as the original distribution and that minimizes the size of the subset. We first formulate the sampling problem with the chi-square test, by using Bayesian networks. Then, a heuristic sampling method is proposed; the method adopts the genetic algorithm to optimize the extracted samples during each iteration by combining a chi-square-test based scoring function and a likelihood-scoring function as the fitness function. The algorithm's performance is tested on four different types of datasets, which are generated based on four Bayesian networks: ASIA, ALARM, HEPAR2, and ANDES. The results show that when the significant difference level, α, is set to 0.05, the proposed algorithm excludes 99.9%, 99.0%, 93.1% and 96.7% of the samples from their original datasets with the size of 60 000 while still satisfying the constraints of the chi-square test. Our method could find much smaller subsets compared with Yang *et al.*'s method [15] under the distribution constraints. Averagely, the subset size obtained by our method is about 47.5% of that obtained by Yang *et al.*'s method [15].

The rest of this paper is organized as follows. In Section 2, we introduce related work. In Section 3, the problem is formulated, and the sampling method is introduced. The experimental results are described in Section 4, and we summarize the paper in the last section.

2 Related Work

A Bayesian network is a probabilistic graphical model [21], which organizes the attributes into a directed graph (DAG). In the graph, each node represents a variable or attribute. Assume there are n variables, i.e., $\{x_1, x_2, ..., x_n\}$; then, there should be n nodes in the graph. A directed edge from x_i to x_j means that the values of x_i can influence x_j. x_i is usually named as a parent node, and x_j is named as a children node. A node may have multiple parent nodes and multiple children nodes. Here, we use $\pi(x_i)$ to represent the parent node set of x_i. For each node, the graph defines a conditional probability table based on the node's parent nodes. Fig.1 illustrates an example of a Bayesian network. In the graph, there are five nodes including $\{A, B, C, D, E\}$. For each node, there is a conditional probability table, which stores the probabilities of each variable's values conditioning on the variable's parent nodes.

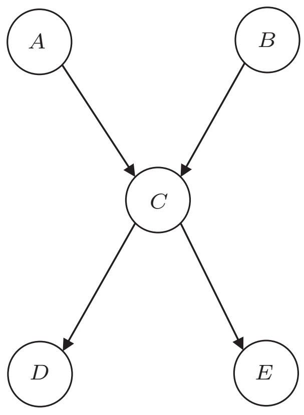
The conditional probability tables can be regarded as displaying the parameters of a Bayesian network. Here, we use θ_{ijk} to encode the parameters, which denotes the probability of the k-th value of node x_i conditioning on the j-th combination of the node's parent nodes $\pi(x_i)$. With these tables, the joint probability distribution can be factorized as the product of all the conditional probability distributions in the network [22], which can be written as (1):

$$P\left(x_1, \ldots, x_n\right) = \prod_{i=1}^{n} P\left(x_i | \pi\left(x_i\right)\right). \tag{1}$$

The joint probability distribution of a dataset can reflect some key characteristics of the dataset. Many machine learning methods are based on the probability distribution [23–25]. In addition, researchers in medicine and biology often focus on a dataset's statistical properties, which also depend on the probability distribution. In this paper, we aim to extract a sampled subset to replace the original whole dataset. The basic idea is to guarantee the joint probability distribution, which is usually hard to obtain. According to (1), if we main-

Parameter	A	$P(A)$
θ_{111}	Yes	0.99
θ_{112}	No	0.01

Parameter	B	$P(B)$
θ_{211}	Yes	0.90
θ_{212}	No	0.10



Parameter	C	A	B	$P(C\|A,B)$
θ_{311}	Yes	Yes	Yes	0.01
θ_{312}	No	Yes	Yes	0.99
θ_{321}	Yes	Yes	No	0.99
θ_{322}	No	Yes	No	0.01
θ_{331}	Yes	No	Yes	0.75
θ_{332}	No	No	Yes	0.25
θ_{341}	Yes	No	No	0.25
θ_{342}	No	No	No	0.75

Parameter	D	C	$P(D\|C)$
θ_{411}	Yes	Yes	0.24
θ_{412}	No	Yes	0.76
θ_{421}	Yes	No	0.36
θ_{422}	No	No	0.64

Parameter	E	C	$P(E\|C)$
θ_{511}	Yes	Yes	0.12
θ_{512}	No	Yes	0.88
θ_{521}	Yes	No	0.25
θ_{522}	No	No	0.75

Fig.1. Bayesian network example. It contains five nodes, i.e., $\{A, B, C, D, E\}$. For each node, there is a conditional probability table, in which $\theta_{ijk} = P\left(x_i = k|\pi\left(x_i\right) = j\right)$. The table displays the probability of the k-th value of node x_i conditioning on the j-th combination of the node's parent nodes $\pi(x_i)$.

tain the conditional probability distributions, then the joint distribution is ensured.

3 Method

In this section, we first formulate the sampling problem to be evaluated by the chi-square test; then, we describe the framework for sampling. Finally, the scoring functions and the detailed procedures are introduced.

3.1 Problem Formulation

Assume M is the size of the original dataset D and D contains n variables. Let B be the corresponding Bayesian network of D. The parameter set of B is $\boldsymbol{\theta}$, which consists of θ_{ijk} ($1 \leqslant i \leqslant n$, $1 \leqslant j \leqslant q_i$, $1 \leqslant k \leqslant r_i$). q_i and r_i represent the number of all combinations of $\pi(x_i)$ and the number of unique values of x_i, respectively. θ_{ijk} denotes the probability of the k-th value of node x_i conditioning on the j-th combination of the node's parent nodes $\pi(x_i)$, which can be determined based on D by $\theta_{ijk} = P(x_i = k|\pi(x_i) = j)$.

In Section 2, we mentioned that we want to maintain the conditional probability distributions when sampling. Therefore, the problem is to choose a subset D' such that the conditional probability distributions of D' and D are sufficiently close. The conditional probability distributions of a given dataset are represented by θ_{ijk}; therefore, the probability distribution test can be conducted on θ_{ijk}. Here, we use the chi-square test to determine the similarity between these two distributions. Thus, the sampling problem is converted to find a subset D' with the minimum size m that satisfies the chi-square test, which is denoted as follows:

$$test(D'; D; i, j) \triangleq \sum_{k=1}^{r_i} \frac{\left(m_{ijk} - \theta_{ijk}\sum_{k=1}^{r_i} m_{ijk}\right)^2}{\theta_{ijk}\sum_{k=1}^{r_i} m_{ijk}} < \chi^2_\alpha(p), \qquad (2)$$

$$test'(D'; D; i) \triangleq \sum_{j=1}^{q_i}\sum_{k=1}^{r_i} \frac{(m_{ijk} - \eta_{ijk}m)^2}{\eta_{ijk}m} < \chi^2_\alpha(p). \qquad (3)$$

In (3), η_{ijk} is the joint distribution of node x_i and its parent nodes $\pi(x_i)$ in dataset D, which can be calculated according to $P(x_i, \pi(x_i))$. m_{ijk} is the number of the k-th value of x_i conditioning on the j-th combination of $\pi(x_i)$ in dataset D'. q_i is the number of all possible combinations of $\pi(x_i)$. r_i is the number of unique values of x_i. α denotes the significant difference level in the chi-square test. p is the degrees of freedom in the chi-square distribution.

These two equations are used for conducting the chi-square test on node x_i. The conditional probabilities are defined on each node; therefore the probability distribution test is performed on each node. For node x_i, there are conditional probability θ_{ijk} and joint probability η_{ijk}, which are tested by (2) and (3), respectively. In these two equations, θ_{ijk} and η_{ijk} are calculated based on dataset D and represent the distribution of D, and m_{ijk} and m are from dataset D' and represent the distribution of D'.

$\chi^2_\alpha(p)$ is the prefixed chi-square test threshold value, which can be achieved with α and p by checking the chi-square distribution table. If the chi-square value is less than this threshold, we can say these two distributions have no significant difference with a $(1-\alpha)$ confidence level. Both equations have p, which denotes the degrees of freedom. It represents the number of variables that are free to vary without influencing the result of the statistics[26,27]. In (2) and (3), there are r_i and q_ir_i variables, respectively. If θ_{ijk} is equal to 0, then variable m_{ijk} should also be 0. These variables should be combined when counting the degrees of freedom. Besides, if all variables except one are determined, then the remained variable cannot vary freely in order to guarantee the result of the statistics. Therefore, the degrees of freedom in (2) and (3) should be $r_i - \sum_{k=1}^{r_i} I(\theta_{ijk} \neq 0) - 1$ and $q_ir_i - \sum_{j=1}^{q_i}\sum_{k=1}^{r_i} I(\theta_{ijk} \neq 0) - 1$, respectively. When θ_{ijk} is equal to 0, the value of function $I()$ is 1; otherwise, the value is 0.

Note that we assume the distribution is faithful here. We apply chi-square tests to guarantee there are no significant distribution differences between the sampled subset and the original dataset for these conditional distributions. If the distribution is unfaithful, we could still guarantee there are no significant differences on these specified conditional distributions which are calculated from the data. Besides, researchers could directly specify their concerned conditional distributions, which could also be maintained when applying our method.

The problem formulated in this paper could be regarded as a generalized subset selection problem. However, there are still some differences. The classical subset selection problem is to choose a subset of at most k variables to minimize the objection function[28], where k is a prefixed value. It is widely used in machine learning to select a subset of variables to achieve a better prediction performance, e.g., Qian *et al.* proposed a POSS approach by employing evolutionary Pareto optimization to obtain a better regression performance[29]. Lately, they accelerated POSS by applying a parallelization strategy[30]. The problem solved in this paper is to find a subset of samples that must pass the distribution test which means (2) and (3) must be satisfied. The distribution test constraints could guarantee there is no significant distribution difference between the sampled subset and the original dataset. Under these constraints, we want to minimize the size of this subset. This is also the reason that we design a distribution-test based scoring function to guide the sampling process. The detailed sampling process will be introduced in Subsection 3.2.

3.2 Heuristic Sampling Algorithm

The main idea of the proposed sampling algorithm is to apply a heuristic strategy to find a subset D' with a minimum size m based on the defined scoring functions. First, D' is initialized to an empty set; then, data samples are gradually added into D' based on the scoring functions. The process is listed as follows:

- *Step* 1: enumerate all the combinations of $\pi(x_i)$ for each node x_i, and encode these combinations;
- *Step* 2: initialize an empty set D', and calculate θ_{ijk} and η_{ijk} based on dataset D and the given Bayesian network structure;
- *Step* 3: apply the genetic algorithm to choose a subset $\Delta D'$ with size $|\Delta D'|$ based on the scoring function values, and add this subset into D';
- *Step* 4: if all the nodes pass the chi-square test, i.e., if the nodes satisfy (2) and (3), then terminate; otherwise go to step 3.

In step 1, we enumerate all the combinations of $\pi(x_i)$ for each node x_i and encode them. These codes are used as j in θ_{ijk} and η_{ijk}. Step 2 initializes D' and calculates parameters θ_{ijk} and η_{ijk}, which represent the distribution of D and will be used for the chi-square test. In step 3, a subset $\Delta D'$ is chosen by the genetic algorithm which mimics the process of natural biological evolution to optimize a given function[31]. The scoring functions for choosing $\Delta D'$ will be described in Subsection 3.3.

This subset will be added to D'. Step 4 is used to determine whether D' satisfies the requirement. If so, then the process is terminated; otherwise, we go to step 3 to continue adding samples into D'.

3.3 Scoring Functions

In this subsection, we introduce how to evaluate a sample or a sample set by defining scoring functions. Here, two scoring functions are used, including

$$L(D,D') = \sum_{i=1}^{n}\left(\sum_{j=1}^{r_i}(I(test(D';D;i,j))) + I(test'(D';D;i))\right), \quad (4)$$

and

$$W(D,D') = \sum_{i=1}^{n}\sum_{j=1}^{q_i}\sum_{k=1}^{r_i} M_{ijk}\log\left(\frac{m_{ijk}}{\sum_{k=1}^{r_i} m_{ijk}}\right), \quad (5)$$

which are derived from the chi-square test and the Dirichlet distribution assumption, respectively. These two scoring functions are used to evaluate the similarity between the distribution of D and that of D'. (4) applies $test(D';D;i,j)$ and $test'(D';D;i)$ to verify whether the j-th conditional probability distribution of x_i and the joint probability distribution of x_i and $\pi(x_i)$ satisfy the chi-square test. The calculations are based on (2) and (3). If the test is passed, the value of $I()$ function equals 0; otherwise, the value equals 1. Therefore, (4) determines how many nodes fail the chi-square test.

In (5), M_{ijk} and m_{ijk} are the number of the k-th value of x_i conditioning on the j-th combination of $\pi(x_i)$ in datasets D and D', respectively. This equation assumes samples in the dataset obey the Dirichlet distribution, which can be derived based on the maximum likelihood assumption. The derivation process is shown below.

Assuming γ' is the Dirichlet distribution parameter of D', the likelihood value $W(D,D') = \log(\mathrm{P}(D|\mathrm{D}')) = \log(\mathrm{P}(D|\gamma'))$. Let us determine the value of γ'.

According to the Dirichlet distribution assumption, the probability of a sample d_l in D' is defined as

$$\mathrm{P}(d_l|\gamma') = \prod_{i=1}^{n}\prod_{j=1}^{q_i}\prod_{k=1}^{r_i}\chi(i,j,k;d_l)^{\gamma'_{ijk}},$$

where

$$\chi(i,j,k;d_l) = \begin{cases} 1, & \text{if } x_i = k, \pi(x_i) = j, \\ 0, & \text{otherwise.} \end{cases}$$

Then, the likelihood function of d_l can be written as

$$\log(\mathrm{P}(d_l|\gamma')) = \sum_{i=1}^{n}\sum_{j=1}^{q_i}\sum_{k=1}^{r_i}\chi(i,j,k;d_l)\log(\gamma'_{ijk}).$$

The likelihood function of D' is (6):

$$\log(\mathrm{P}(D'|\gamma')) = \sum_{i=1}^{n}\sum_{j=1}^{q_i}\sum_{k=1}^{r_i} m_{ijk}\log\left(\gamma'_{ijk}\right). \quad (6)$$

Based on Gibb's inequality, if $\mathrm{P}(x)$ and $\mathrm{Q}(x)$ are two probability distributions over the same domain, then $\sum_x \mathrm{P}(x)\log(\mathrm{Q}(x)) \leqslant \sum_x \mathrm{P}(x)\log(\mathrm{P}(x))$. Therefore, to maximize the likelihood function, i.e., (6), γ'_{ijk} must satisfy

$$\gamma'_{ijk} = \frac{m_{ijk}}{\sum_{k=1}^{r_i} m_{ijk}}.$$

Therefore,

$$\begin{aligned} & W(D, D') \\ &= \log(\mathrm{P}(D|\gamma')) \\ &= \sum_{i=1}^{n}\sum_{j=1}^{q_i}\sum_{k=1}^{r_i} M_{ijk}\log\left(\gamma'_{ijk}\right) \\ &= \sum_{i=1}^{n}\sum_{j=1}^{q_i}\sum_{k=1}^{r_i} M_{ijk}\log\left(\frac{m_{ijk}}{\sum_{k=1}^{r_i} m_{ijk}}\right). \end{aligned}$$

According to Gibb's inequality, $W(D, D')$ achieves the maximum value only if γ'_{ijk} satisfies

$$\gamma'_{ijk} = \theta_{ijk} = \frac{M_{ijk}}{\sum_{k=1}^{r_i} M_{ijk}},$$

thus indicating that the distributions of D and D' are the same. This indication is why we also apply (5) as a scoring function.

In each iteration, we add a subset $\Delta D'$ into D'. The evaluation is conducted for $\Delta D'$, which is denoted as follows:

$$\begin{aligned} & L(D, D', \Delta D') \\ &= \sum_{i=1}^{n}\left(\sum_{j=1}^{r_i}(I(test(D' \cup \Delta D'; D; i, j))) + \right. \\ & \left. I(test'(D' \cup \Delta D'; D; i))\right), \end{aligned} \quad (7)$$

$$\begin{aligned} & W(D, D', \Delta D') \\ &= \sum_{i=1}^{n}\sum_{j=1}^{q_i}\sum_{k=1}^{r_i} M_{ijk}\log\left(\frac{m_{ijk} + m^*_{ijk}}{m_{ijk}} \times \right. \\ & \left. \frac{\sum_{k=1}^{r_i} m_{ijk}}{\sum_{k=1}^{r_i}(m_{ijk} + m^*_{ijk})}\right). \end{aligned} \quad (8)$$

(8) is defined as $W(D, D', \Delta D') = \log(\mathrm{P}(D|D' \cup \Delta D')) - \log(\mathrm{P}(D|D'))$. Based on (5), we can obtain (8). m^*_{ijk} is the number of the k-th value of x_i conditioning on the j-th combination of $\pi(x_i)$ in dataset $\Delta D'$.

(7) denotes how many nodes fail the chi-square test after adding $\Delta D'$ into D', while (8) reflects the increase in scoring values after the adding operation. Therefore, we should find $\Delta D'$ with a smaller value of $L(D, D', \Delta D')$ based on (7) and choose $\Delta D'$ with a larger value of $W(D, D', \Delta D')$ based on (8). When comparing scoring function values in the genetic algorithm, we first compare the values of $L(D, D', \Delta D')$. If the values of $L(D, D', \Delta D')$ are the same, we then compare the values of $W(D, D', \Delta D')$. In addition, to accelerate the sampling process, we count only those nodes that fail the chi-square test in the calculation of (8).

3.4 Encoding

In the sampling algorithm, we need to calculate all the values of m_{ijk} for the scoring functions. The value of m_{ijk} will change if we add a sample. Therefore, we need to compute these values repetitively during each iteration, and this computation is quite time consuming. Actually, we need to calculate only the change of m_{ijk}, thereby reducing the time complexity. Here, we apply an encoding strategy to fulfill this task.

In m_{ijk}, i denotes the i-th variable, j is the serial number of the combination of the parent nodes $\pi(x_i)$ of x_i, and k is the value of x_i. For a given sample d, i and k can be easily determined. The most difficult part is determining j. Assume x_i has $|\pi(x_i)|$ parent nodes, i.e., $\pi(x_i) = \{x_i^1, x_i^2, ..., x_i^{|\pi(x_i)|}\}$. Assume there are r_i^l possible values for parent node x_i^l. In sample d, the values of $\pi(x_i)$ are $\{k_i^1, k_i^2, ..., k_i^{|\pi(x_i)|}\}$. Then, the combination serial number j can be determined by (9):

$$j = (((k_i^1 r_i^2) + k_i^2) r_i^3 + \cdots + k_i^{|\pi(x_i)|-1}) r_i^{|\pi(x_i)|} + k_i^{|\pi(x_i)|}. \quad (9)$$

For each sample, the combination serial number of parent nodes for each node x_i can be calculated during preprocessing. This value will not change during the algorithm. Therefore, we can store these values in an array. When updating m_{ijk}, we can check only this array to accomplish the task.

3.5 Time Complexity Analysis

In this subsection, we list the pseudocodes of our algorithm and then analyze its time complexity. Algorithm 1 shows the main process of the sampling algorithm.

In Algorithm 1, line 1 initializes D' to an empty set. Line 2 initializes an array $\boldsymbol{m}$ with 0; the array is used to record the distribution of D'. Line 3 calls procedure *preProcess* to determine the distribution of D and the encoding values of the parent nodes of each node x_i for each sample. The distribution is recorded by $\boldsymbol{\theta}$ and $\boldsymbol{\eta}$, and the encoding values are recorded by $\boldsymbol{idxJ}$. The loop of lines 4–13 is the main sampling process. In each iteration, the algorithm extracts at least $|\Delta D'|$ samples from D; therefore there are at most $M/(|\Delta D'|)$ iterations. Line 5 applies the genetic algorithm to select $|\Delta D'|$ samples based on the scoring functions; thus, the values of the score functions are used as the fitness values in the genetic algorithm. Then, the distribution of $\Delta D'$ is calculated in line 6 by calling procedure *distCal*. Line 7 calls procedure *scoreCal* to compute the chi-square test scoring value and the likelihood scoring value, which are recorded by *chiScore* and *dScore*, respectively. Lines 8 and 9 are used to merge $\Delta D'$ and D'. The chi-square test value reflects how many nodes fail the distribution test. Therefore, if this value is equal to 0, then we can terminate the algorithm, as we have already achieved the required subset. The condition and steps for terminating the algorithm are in lines 10–12.

Algorithm 1. Sampling Algorithm

```
   Input: dataset $D$, Bayesian network $B$
   Output: subset $D'$
1  $D' = \emptyset$;
2  $\boldsymbol{m} = \boldsymbol{0}$;
3  ($\boldsymbol{idxJ}$, $\boldsymbol{\theta}$, $\boldsymbol{\eta}$)=$preProcess(D, B)$;
4  for $h = 0; h < M/(|\Delta D'|); h++$ do
5      $\Delta D' = GA(\boldsymbol{idxJ}, \boldsymbol{m}, \boldsymbol{\theta}, \boldsymbol{\eta})$;
6      $\boldsymbol{m}^* = distCal(\Delta D', \boldsymbol{idxJ})$;
7      $(chiScore, dScore) = scoreCal(\boldsymbol{m}, \boldsymbol{m}^*, \boldsymbol{\theta}, \boldsymbol{\eta})$;
8      $\boldsymbol{m} = \boldsymbol{m} + \boldsymbol{m}^*$;
9      $D' = D' \cup \Delta D'$; $D = D - \Delta D'$;
10     if $chiScore = 0$ then
11         break;
12     end
13 end
```

Algorithm 2 describes the detailed procedure of preprocessing. The main purpose of preprocessing is to calculate the distribution parameters $\boldsymbol{\theta}$ and $\boldsymbol{\eta}$ of D and to encode the values of the combinations of the parent nodes of x_i for each sample. $idxJ(h,i)$ records the encoding value of the parent nodes of x_i in the h-th sample. These values will be used to calculate m_{ijk} and θ_{ijk}. In Algorithm 2, lines 1–10 calculate $idxJ(h,i)$ based on (9). In line 1, h indicates the sample index number, which ranges from 0 to $|D|$. In line 2, i represents the variable index number, which ranges from 0 to n. In line 3, hh is used to mark x_i's hh-th parent node. The value and the range of this parent node are denoted by k_i^{hh} and r_i^{hh}, respectively. $totalC(i)$ records the total combination number of the parent nodes of x_i. $d(h,i)$ is the value of x_i in the h-th sample. Lines 11–27 calculate $\boldsymbol{\theta}$ and $\boldsymbol{\eta}$ for dataset D. In Algorithm 2, lines 4, 5, 7 and 8 can be accomplished in $O(1)$ time. Assume the largest values of $totalC(i)$ and r_i are q and r, respectively. $|\pi(x_i)|$ in line 3 should be smaller than $totalC(i)$. Therefore, lines 1–10 take $O(|D|nq)$ time. Lines 11–15 take $O(|D|n)$ time. Lines 16–27 take $O(nqr)$ time. The *preProcess* procedure takes $O(|D|nq + |D|n + nqr) = O(|D|nq + nqr)$ time.

Algorithm 2. *preProcess*

```
   Input: dataset $D$, Bayesian network $B$
   Output: $\boldsymbol{idxJ}$, $\boldsymbol{\theta}$, $\boldsymbol{\eta}$
1  for $h = 0; h < |D|; h++$ do
2      for $i = 0; i < n; i++$ do
3          for $hh = 1; hh \leqslant |\pi(x_i)| - 1; hh++$ do
4              $idxJ(h,i) = (idxJ(h,i) + k_i^{hh}) r_i^{hh+1}$;
5              $totalC(i) = totalC(i) \times r_i^{hh}$;
6          end
7          $idxJ(h,i) = idxJ(h,i) + k_i^{|\pi(x_i)|}$;
8          $totalC(i) = totalC(i) \times r_i^{|\pi(x_i)|} - 1$;
9      end
10 end
11 for $h = 0; h < |D|; h++$ do
12     for $i = 0; i < n; i++$ do
13         $\theta_{i,\, idxJ(h,i),\, d(h,i)}{++}$;
14     end
15 end
16 for $i = 0; i < n; i++$ do
17     for $j = 0; j \leqslant totalC(i); j++$ do
18         $sum = 0$;
19         for $k = 0; k < r_i; k++$ do
20             $sum = sum + \theta_{ijk}$;
21         end
22         for $k = 0; k < r_i; k++$ do
23             $\eta_{ijk} = \theta_{ijk}/|D|$;
24             $\theta_{ijk} = \theta_{ijk}/sum$;
25         end
26     end
27 end
```

Algorithm 3 applies the genetic algorithm to find a subset $\Delta D'$ with size $|\Delta D'|$. In line 1, the algorithm randomly chooses C subsets with size $|\Delta D'|$ from the remained samples to form an initial population P and encodes genes with the serial numbers of the selected samples in each subset. Thus, a subset corresponds to a gene in population P and P contains C genes. These genes have the same length $|\Delta D'|$. The loop of lines 2–16 executes L iterations to optimize these genes. Lines 3–6 call procedures *distCal* and *scoreCal* to calculate the scoring values for each gene. Line 7 combines these two scoring values, and then line 8 performs score scaling based on the rank of each gene in the sorted population P. Lines 9 and 10 determine the number of parents that will be used to generate the

Algorithm 3. *GA*

```
Input: $\boldsymbol{idxJ}$, $\boldsymbol{m}$, $\boldsymbol{\theta}$, $\boldsymbol{\eta}$
Output: $\Delta D'$
1  Initialize the population set $P$ with size $C$;
2  for $ii = 0; ii < L; ii++$ do
3      for each gene $P_{jj}$ in $P$ do
4          $\boldsymbol{m}^*$=$distCal(P_{jj}, \boldsymbol{idxJ})$;
5          $(chiScore[jj], dScore[jj])$=$scoreCal(\boldsymbol{m}, \boldsymbol{m}^*, \boldsymbol{\theta}, \boldsymbol{\eta})$;
6      end
7      $Score = chiScore \times e^{10} - dScore$;
8      $fitValue$=$FitScaling(Score)$;
9      $nCrParent = 2(C - nElite) \times CrRate$;
10     $nMuParent = C - nElite - nCrParent/2$;
11     $pSet$=$Select(nCrParent, nMuParent, P, fitValue)$;
12     $cChild$=$Crossover(pSet, nCrParent)$;
13     $mChild$=$Mutate(pSet, nMuParent)$;
14     $eChild$=$FindBest(P, nElite)$;
15     $P = eChild \cup cChild \cup mChild$;
16 end
17 Update $Score$ for $P$;
18 $\Delta D'$=$FindBest(P, 1)$;
```

next population. $CrRate$ is set to 0.8 and represents the crossover rate. $nElite$ is set to $0.05C$ and represents the number of genes in population P that will survive to the next generation. Line 11 applies a stochastic uniform strategy to select $nCrParent + nMuParent$ genes from population P based on $fitValue$. Line 12 applies a scattered crossover strategy to generate $0.5nCrParent$ new genes using $nCrParent$ genes in the selected $pSet$. Line 13 applies a uniform mutation strategy to generate $nMuParent$ new genes with mutation rate 0.01 using $nMuParent$ genes in the selected $pSet$. Line 14 chooses the best $nElite$ genes from P based on $Score$. After the loop, line 17 reruns lines 3–7 to update $Score$ for P, and then line 18 chooses the best gene from the finial population P. The procedures $FitScaling$, $Select$, $Crossover$, and $Mutate$ are adopted from the Matlab genetic algorithm toolbox; therefore we do not list the detailed pseudocodes here. Procedure $FindBest$ can be accomplished by sorting the population P. Procedures $disCal$ and $scoreCal$ take $O(|\Delta D'|n)$ time and $O(nqr)$ time, respectively, and the population P contains C genes; therefore lines 3–6 take $O(Cnqr)$ time. Procedure $FitScaling$ in line 8 needs $O(C\log C)$ time as it needs to sort the genes in P. Procedure $Select$ in line 11 takes $O(nCrParent + nMuParent)$ time. Procedures $Crossover$ and $Mutate$ in line 12 and line 13 need $O(0.5nCrParent|\Delta D'|)$ time and $O(nMuParent|\Delta D'|)$ time, respectively, as both procedures generate new genes with size $|\Delta D'|$. Procedure $FindBest$ takes $O(C\log C)$ time as it just needs to sort the population P. Therefore, lines 2–16 need $O(L(Cnqr + C\log C + (nCrParent + nMuParent)|\Delta D'|))$ time. Line 17 reruns lines 3–7 and needs $O(Cnqr)$ time. Line 18 takes $O(C\log C)$ time. Experimentally, L, C, $|\Delta D'|$, $CrRate$ and $nElite$ are set to constant values; therefore the time complexity of Algorithm 3 can be simplified to $O(nqr)$.

Algorithm 4 calculates $\boldsymbol{m}^*$ for dataset $\Delta D'$, which represents the distribution and will be used in the calculation of score values. m^*_{ijk} is the number of the k-th value of node x_i conditioning on the j-th combination of the node's parent nodes $\pi(x_i)$ in $\Delta D'$. Therefore, the main loop of this procedure is to check each sample in $\Delta D'$ and update m^*_{ijk} by finding the corresponding value. As the encoding value j is indexed based on the index numbers of samples in D, line 3 finds this index number in D for the ii-th sample of $\Delta D'$. With this number, we can find the encoding value j of the parent nodes of x_i for sample ii. $d(h, i)$ is x_i's value in sample ii. In this procedure, lines 3 and 4 take $O(1)$ time. Thus, the whole procedure takes $O(|\Delta D'|n)$ time.

Algorithm 4. *distCal*

```
Input: dataset $\Delta D'$, $\boldsymbol{idxJ}$
Output: $\boldsymbol{m}^*$
1  for $ii = 0; ii < |\Delta D'|; ii++$ do
2      for $i = 0; i < n; i++$ do
3          $h = index(ii)$;
4          $m^*_{i,\,idxJ(h,i),\,d(h,i)} = m^*_{i,\,idxJ(h,i),\,d(h,i)} + 1$;
5      end
6  end
```

Algorithm 5 presents the detailed procedures for calculating scoring functions based on (7) and (8). The inputs $\boldsymbol{m}$ and $\boldsymbol{m}^*$ denote the distribution of D' and $\Delta D'$, respectively. $\boldsymbol{\theta}$ and $\boldsymbol{\eta}$ represent the distribution of D. Line 1 conducts the summation calculation to obtain the distribution of the joint set of D' and $\Delta D'$. The loop of lines 2–27 calculates the chi-square test scoring value. This loop is a nested loop, where i ranges from 0 to n, j ranges from 0 to q_i, and k ranges from 0 to r_i. n, q_i, and r_i denote the number of variables, the total combinatorial number of the parent nodes of x_i, and the number of unique values of x_i, respectively. There are two chi-square tests (i.e., (2) and (3)) in the problem formulation. $chiSquare1$ records the first test, and $chiSquare2$ records the second test. To accelerate the sampling process, the likelihood score values count only the variables that fail the chi-square test. These variables are stored in array $node$. Lines 16–18 and lines 23–25 conduct this operation. Then, lines 28–39 compute the likelihood score value on these variables. To be inconsistent with Algorithm 2, we assume the maximum-value of q_i and r_i is q and r, respectively.

The arithmetic operations in Algorithm 4 take $O(1)$ time. Line 17 also takes $O(1)$ time. Consequently, lines 5–19 take $O(r)$ time, and lines 2–27 take $O(nqr)$ time. There are at most n variables in *node*; therefore lines 28–39 take $O(nqr)$ time. The *scoreCal* procedure takes $O(nqr)$ time in total.

Algorithm 5. *scoreCal*

```
   Input: $\boldsymbol{m}$, $\boldsymbol{m^*}$, $\boldsymbol{\theta}$, $\boldsymbol{\eta}$, $\boldsymbol{idxJ}$
   Output: $chiScore$, $dScore$
1  $\hat{\boldsymbol{m}} = \boldsymbol{m} + \boldsymbol{m^*}$;
2  for $i = 0; i < n; i++$ do
3      $chiSquare2 = 0$;
4      for $j = 0; j < q_i; j++$ do
5          $sum = 0$;   $chiSquare1 = 0$;
6          for $k = 0; k < r_i; k++$ do
7              $sum = sum + \hat{m}_{ijk}$;
8          end
9          $size = |D'| + |\Delta D'|$;
10         for $k = 0; k < r_i; k++$ do
11             $chiSquare1 += \frac{(\hat{m}_{ijk} - \theta_{ijk} \times sum)^2}{\theta_{ijk} \times sum}$;
12             $chiSquare2 += \frac{(\hat{m}_{ijk} - \eta_{ijk} \times size)^2}{\eta_{ijk} \times size}$;
13         end
14         if $chiSquare1 \geqslant \chi^2_\alpha(p_1)$ then
15             $chiScore += 1$;
16             if $vis(i) = 0$ then
17                 $node.pushback(i)$;   $vis(i) = 1$;
18             end
19         end
20     end
21     if $chiSquare2 \geqslant \chi^2_\alpha(p_2)$ then
22         $chiScore += 1$;
23         if $vis(i) = 0$ then
24             $node.pushback(i)$;   $vis(i) = 1$;
25         end
26     end
27 end
28 for $ii = 0; ii < node.size; ii++$ do
29     $i = \mathrm{node}(ii)$;
30     for $j = 0; j < q_i; j++$ do
31         $sum_m = sum = 0$;
32         for $k = 0; k < r_i; k++$ do
33             $sum_m = sum_m + m_{ijk}$;
               $sum = sum + \hat{m}_{ijk}$;
34         end
35         for $k = 0; k < r_i; k++$ do
36             $dScore += \theta_{ijk}|D|\log(\frac{\hat{m}_{ijk} \times sum_m}{m_{ijk} \times sum})$;
37         end
38     end
39 end
```

Now, we can analyze the whole time complexity. Line 3 calls the *preProcess* procedure, which takes $O(|D|nq + nqr)$ time. Line 5 applies the *GA* procedure to generate a subset with size $|\Delta D'|$, which needs $O(nqr)$ time. Lines 6 and 7 call the *distCal* and *scoreCal* procedures, which take $O(|\Delta D'|n)$ time and $O(nqr)$ time, respectively. $\boldsymbol{m}$ is an array with size nqr; therefore line 8 takes $O(nqr)$ time. Line 9 takes $O(|\Delta D'|)$ time. Therefore, lines 5–12 take $O(nqr)$ time. The whole sampling algorithm takes $O(|D|nq + nqr + |D|nqr) = O(|D|nqr)$ time.

4 Experiment

The distribution tests are conducted according to the structures of Bayesian networks. To validate the performance of the proposed method, four Bayesian networks are adopted in the experiments: ASIA, ALARM, HEPAR2, and ANDES.

• *ASIA*. The ASIA network [32] is a small network that is used for diagnosing chest diseases, including tuberculosis, lung cancer, and bronchitis. This network contains eight nodes and eight edges. Each node corresponds to a binary variable.

• *ALARM*. The ALARM network [33] is a medium network that is also used for disease diagnosis. This network consists of 37 nodes and 46 edges. The number of unique values for each variable can be 2, 3 or 4.

• *HEPAR2*. The HEPAR2 network [34] is a large network that is used for the diagnosis of liver disorders. The network consists of 70 nodes and 123 edges.

• *ANDES*. ANDES [35] is a very large network that is used in an intelligent tutoring system. The network consists of 223 nodes and 338 edges.

For each network, a Gibbs sampler is applied to generate three datasets of different sizes, including 20 000, 40 000 and 60 000. In the genetic algorithm, the iteration number is set to 500, and the population size is set to 200. Other settings including crossover rate, mutation rate, and number of genes that will survive to the next generation take the default values of Matlab genetic algorithm toolbox. To choose a suitable $|\Delta D'|$, i.e., the number of samples extracted in each iteration, we let $|\Delta D'|$ range from 1 to 50 and compare the size of the sampled subset on the ALARM datasets. Fig.2 illustrates this result. Generally, the method achieves the best results when $|\Delta D'| = 10$; therefore, we set $|\Delta D'| = 10$ in the following experiments.

In statistics, the significant difference level is usually set to 0.05 or 0.01 to denote whether to accept the null hypothesis. Here, the null hypothesis is that the distributions of the sampled subset and the original dataset are the same. Therefore, we set the significant difference level, α, to 0.05 and 0.01 to test our method. Table 1 summarizes the comparison results between our method and Yang *et al.*'s method [15] in terms of subset size, which is the size of the subset that is extracted from the original dataset to meet the chi-square test requirement. In Table 1, 20 000, 40 000 and 60 000 are the sizes of datasets, so are in Tables 2–4.

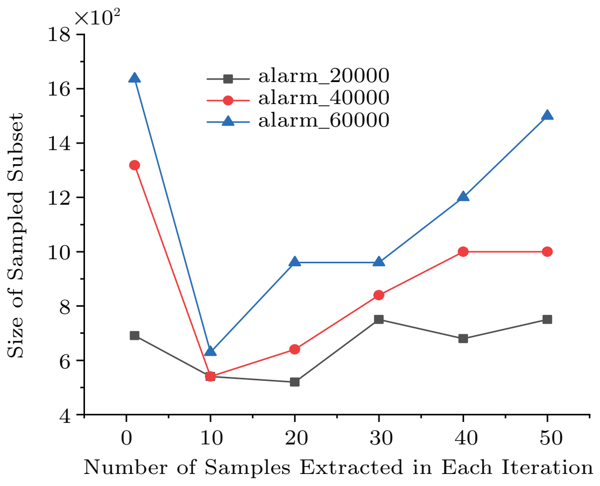


Fig. 2. Comparison of the size of the sampled subsets when $|\Delta D'|$ ranges from 1 to 50 on three ALARM datasets, where alarm_20000, alarm_40000 and alarm_60000 represent the datasets with the size of 20 000, 40 000 and 60 000, respectively.

We find that our method excludes most of the samples from the original datasets while still maintaining the distribution. For the datasets with the size of 60 000, we can exclude 99.9%, 99.0%, 93.1% and 96.7% of the samples from the original datasets for ASIA, ALARM, HEPAR2, and ANDES datasets, respectively, when setting α to 0.05. Averagely, the subset size obtained by our method is about 47.5% of that obtained by Yang *et al.*'s method [15], which validates that our method could extract much smaller subsets than Yang *et al.*'s method [15] to meet the distribution requirement. In addition, the size of the sampled subset becomes larger as the structure of the Bayesian network becomes more complex because the distribution depends on the Bayesian network structure. As the structure becomes more complex, the distribution of the original dataset also becomes more complex. Consequently, more samples are needed to form the complex distribution. Similarly, as the size of the original dataset becomes larger, small probabilities are more likely to happen; consequently, the distribution becomes more complex. Therefore, more samples are needed to satisfy the requirement. For the ASIA network, the sample sizes are the same for different sizes of datasets because the network is small and the distribution is simple. It is very easy to satisfy (2) and (3). The sample sizes where $\alpha = 0.01$ are generally smaller than the sample sizes where $\alpha = 0.05$ for the same dataset because when $\alpha = 0.01$, more differences in the distribution are allowed than when $\alpha = 0.05$.

We also compare our method with random sampling in terms of the average distribution difference and ratio of satisfied distribution tests when letting the two methods extract the same number of samples. As the subset extracted by Yang *et al.*'s method [15] can also satisfy the distribution tests, we do not compare our method with their method in this experiment. The average distribution difference is calculated based on

$$delta = \frac{1}{num(\theta_{ijk})}\sum_{i=1}^{n}\sum_{j=1}^{q_i}\sum_{k=1}^{r_i}\left|\frac{m_{ijk}}{\sum_{k=1}^{r_i} m_{ijk}} - \theta_{ijk}\right|,$$

where $num(\theta_{ijk})$ is the total number of θ_{ijk}. This value reflects the distribution difference between the sampled subset and the original dataset. The ratio of satisfied distribution tests denotes the percentage of distributions that pass the chi-square test. We let the random sampling method extract the same number of samples as our method. As the random sampling method is a probability sampling method, the results are not identical for different executions. The results of random sampling are averaged from 1 000 runs in our experiments.

Table 2 shows the comparison results between our method and random sampling when $\alpha = 0.05$. As our method will not terminate until all the distribution tests are satisfied, the ratios of satisfied distribution

Table 1. Comparison of Experimental Results Between Our Method and Yang *et al.*'s Method [15] on 3 Different Sizes of Datasets Based on 4 Bayesian Networks in Terms of Sampled Subset Size When α Is Set to 0.05 and 0.01

Dataset	$\alpha = 0.05$						$\alpha = 0.01$					
	20 000		40 000		60 000		20 000		40 000		60 000	
	Yang *et al.*'s [15]	Ours	Yang *et al.*'s [15]	Ours	Yang *et al.*'s [15]	Ours	Yang *et al.*'s [15]	Ours	Yang *et al.*'s [15]	Ours	Yang *et al.*'s [15]	Ours
ASIA	100	60	300	60	300	60	100	50	100	50	100	60
ALARM	1 200	540	1 400	540	1 500	610	1 200	510	1 200	530	1 400	530
HEPAR2	4 100	3 010	6 100	3 640	9 100	4 160	3 700	2 800	5 900	3 210	8 700	3 820
ANDES	2 200	1 110	3 200	1 210	5 500	2 010	1 600	1 010	2 500	990	3 700	1 890

Table 2. Comparison of Experimental Results Between Our Method and Random Sampling in Terms of the Average Distribution Difference and Ratio of Satisfied Distribution Tests When $\alpha = 0.05$

Dataset	Random Sampling						Our Heuristic Sampling					
	20 000		40 000		60 000		20 000		40 000		60 000	
	delta	*ratio*	*delta*	*ratio*	*delta*	*ratio*	*delta*	*ratio*	*delta*	*ratio*	*delta*	*ratio*
ASIA	0.22	84.00	0.22	83.06	0.24	82.56	0.01	100	0.01	100	0.01	100
ALARM	0.28	77.02	0.29	76.32	0.31	74.82	0.03	100	0.04	100	0.04	100
HEPAR2	0.32	83.44	0.34	81.90	0.35	80.71	0.05	100	0.06	100	0.06	100
ANDES	0.10	94.15	0.10	93.85	0.08	94.36	0.02	100	0.02	100	0.03	100

Note: *delta* and *ratio* denote the average difference in distribution and the ratio of distributions that pass the chi-square test, respectively.

tests of our method are all 100%. However, the sampled subsets obtained by the random sampling method does not pass all the distribution tests. Approximately 16.8%, 23.9%, 18.0%, and 5.9% of the distribution tests fail the chi-square tests for these four types of datasets. In addition, the average distribution difference of our method is much smaller than that of random sampling. The average distribution difference of our method is approximately 0.03, while the average distribution difference of random sampling is approximately 0.24. These findings validate the effectiveness of our sampling algorithm. Table 3 summarizes the comparison results when $\alpha = 0.01$, and it shows findings similar to Table 2.

Furthermore, we conduct the comparison of running time (in seconds) between our method and Yang *et al.*'s method [15]. Both methods are implemented with Matlab and executed on a Windows server with Intel® Xeon E3-1231v3 3.4 GHz CPU and 16 G RAM. Table 4 shows the results. Our method costs a bit longer time than Yang *et al.*'s method [15]. This is because our method applies the genetic algorithm to optimize the selected subset in each iteration. However, the time complexity of both methods is $O(|D|nqr)$. We can find that the running time of these two methods becomes close when the dataset size, i.e., $|D|$, becomes larger, e.g., 60 000. On some datasets, our method needs shorter time than Yang *et al.*'s method [15], and this may be because the number of iterations of our method is much smaller than that of Yang *et al.*'s method [15] on these datasets.

As our method is heuristic, the samples extracted in each iteration may not be good in the whole sampled subset, i.e., the results are not optimal. To test the quality of the samples in the results, we randomly choose four samples in the ALARM dataset with the size of 40 000 to check the ranks during each iteration, where sample d_l's rank is defined as the serial number of d_l after sorting the samples in D' according to the scoring values. The 3rd sample, the 89th sample, the 413th sample and the 632nd sample are chosen. If a sample d_l is an optimal sample, then its rank in D' should be stable, i.e., when adding new samples into D', these samples' scores should be worse than that of d_l, and d_l's rank should not change much. Fig.3 shows this result. The scoring values are calculated based on (8). We find that although the ranks change during each iteration, the changes are few. The ranks are generally stable, thereby validating the effectiveness of applying genetic algorithm to optimize the samples in each iteration. The subset chosen in each iteration by Yang *et*

Table 3. Comparison of Experimental Results Between Our Method and Random Sampling in Terms of Average Distribution Difference and Ratio of Satisfied Distribution Tests when $\alpha = 0.01$

Dataset	Random Sampling						Our Heuristic Sampling					
	20 000		40 000		60 000		20 000		40 000		60 000	
	delta	*ratio*	*delta*	*ratio*	*delta*	*ratio*	*delta*	*ratio*	*delta*	*ratio*	*delta*	*ratio*
ASIA	0.25	83.12	0.23	83.11	0.25	82.56	0.01	100	0.02	100	0.03	100
ALARM	0.29	79.25	0.29	77.84	0.30	76.33	0.04	100	0.05	100	0.06	100
HEPAR2	0.32	85.41	0.35	83.57	0.36	82.92	0.07	100	0.08	100	0.08	100
ANDES	0.10	96.49	0.11	96.28	0.08	97.17	0.03	100	0.04	100	0.05	100

Note: *delta* and *ratio* denote the average difference in distribution and the ratio of distributions that pass the chi-square test, respectively.

al.'s method[15] is randomly sampled; therefore some samples may not be good for the final subset to pass the distribution test. Although they eliminate some samples with worse scoring values in each iteration, their method could still not avoid these samples. The samples selected in each iteration by our new method are more reasonable. This is also the reason why our method could reduce the size of the final sampled subset.

Table 4. Comparison of Running Time (in Seconds) Between Our Method and Yang *et al.*'s Method[15] on 3 Different Sizes of Datasets When $\alpha = 0.05$

Dataset	Yang *et al.*'s Method[15]			Our Heuristic Sampling		
	20 000	40 000	60 000	20 000	40 000	60 000
ASIA	7	22	25	3	3	4
ALARM	75	102	259	118	115	121
HEPAR2	259	632	1 401	1 643	1 725	1 826
ANDES	312	564	1 101	1 312	1 453	1 610

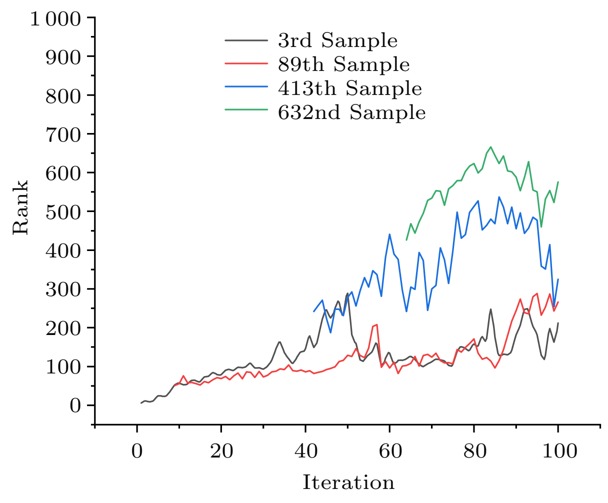


Fig.3. Ranks of four random samples in sampled subsets based on scoring functions.

5 Conclusions

In this paper, we formulated a sampling problem that requires finding a minimum subset while maintaining the probability distribution as verified by the chi-square test. By decomposing the joint distribution into conditional probabilities based on Bayesian networks, we proposed a heuristic sampling method to solve this problem. Our method applies a genetic algorithm to optimize the subset during each iteration and combines a chi-square-test based scoring function and a likelihood-based scoring function as the fitness function. Experiments on four different types of datasets with the size of 60 000 showed that our algorithm can exclude 99.9%, 99.0%, 93.1% and 96.7% of the samples based on the ASIA, ALARM, HEPAR2, and ANDES networks, respectively, when the significant difference level, α, is set to 0.05. Averagely, the subset size obtained by our method is about 47.5% of that obtained by Yang *et al.*'s method[15]. In addition, when sampling subsets of the same size, the average distribution difference of our method is approximately 0.03, which is much smaller than that of the random sampling, whose average distribution difference is approximately 0.24.

In the future, we will try to determine the theoretical bound of the size of the subset that can satisfy the chi-square tests.

References

[1] Goodhart C A E, O'Hara M. High frequency data in financial markets: Issues and applications. *Journal of Empirical Finance*, 1997, 4(2/3): 73-114. DOI: 10.1016/S0927-5398(97)00003-0.

[2] Lohr S L. Sampling: Design and Analysis (2nd edition). CRC Press, 2019. DOI: 10.1201/9780429296284.

[3] Yates F. Systematic sampling. *Philosophical Transactions of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 1948, 241(834): 345-377. DOI: 10.1098/rsta.1948.0023.

[4] Neyman J. On the two different aspects of the representative method: The method of stratified sampling and the method of purposive selection. *Journal of the Royal Statistical Society*, 1934, 97(4): 558-625. DOI: 10.2307/2342192.

[5] Rand W M. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 1971, 66(336): 846-850. DOI: 10.2307/2284239.

[6] Aljalbout E, Golkov V, Siddiqui Y *et al.* Clustering with deep learning: Taxonomy and new methods. arXiv:1801.07648, http://export.arxiv.org/abs/1801.07648, March 2020.

[7] Goodman L A. Snowball sampling. *The Annals of Mathematical Statistics*, 1961, 32(1): 148-170. DOI: 10.1214/aoms/1177705148.

[8] Emerson R W. Convenience sampling, random sampling, and snowball sampling: How does sampling affect the validity of research? *Journal of Visual Impairment & Blindness*, 2015, 109(2): 164-168. DOI: 10.1177/0145482X1510900215.

[9] Saar-Tsechansky M, Provost F. Active sampling for class probability estimation and ranking. *Machine Learning*, 2004, 54(2): 153-178. DOI: 10.1023/B:MACH.0000011806.12374.c3.

[10] Dasgupta S, Hsu D. Hierarchical sampling for active learning. In *Proc. the 25th International Conference on Machine Learning*, June 2008, pp.208-215. DOI: 10.1145/1390156.1390183.

[11] Zhang H, Lin J, Cormack G V, Smucker M D. Sampling strategies and active learning for volume estimation. In *Proc. the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, July 2016, pp.981-984. DOI: 10.1145/2911451.2914685.

[12] Silva J, Ribeiro B, Sung A H. Finding the critical sampling of big datasets. In *Proc. the Computing Frontiers Conference*, May 2017, pp.355-360. DOI: 10.1145/3075564.3078886.

[13] Alwosheel A, Van Cranenburgh S, Chorus C G. Is your dataset big enough? Sample size requirements when using artificial neural networks for discrete choice analysis. *Journal of Choice Modelling*, 2018, 28: 167-182. DOI: 10.1016/j.jocm.2018.07.002.

[14] Wang A, An N, Chen G, Liu J, Alterovitz G. Subtype dependent biomarker identification and tumor classification from gene expression profiles. *Knowledge-Based Systems*, 2018, 146: 104-117. DOI: 10.1016/j.knosys.2018.01.025.

[15] Yang J, Wang J, Cheng W, Li L. Sampling to maintain approximate probability distribution under chi-square test. In *Proc. the 37th National Conference of Theoretical Computer Science*, August 2019, pp.29-45. DOI: 10.1007/978-981-15-0105-0_3.

[16] Paxton P, Curran P J, Bollen K A *et al.* Monte Carlo experiments: Design and implementation. *Structural Equation Modeling*, 2001, 8(2): 287-312. DOI: 10.1207/S15328007SEM0802_7.

[17] Gilks W R, Richardson S, Spiegelhalter D. Markov Chain Monte Carlo in Practice (1st edition). Chapman and Hall/CRC, 1996.

[18] Wu S, Angelikopoulos P, Papadimitriou C *et al.* Bayesian annealed sequential importance sampling: An unbiased version of transitional Markov chain Monte Carlo. *ASCE-ASME Journal of Risk and Uncertainty in Engineering Systems, Part B: Mechanical Engineering*, 2018, 4(1): Article No. 011008. DOI: 10.1115/1.4037450.

[19] George E I, McCulloch R E. Variable selection via Gibbs sampling. *Journal of the American Statistical Association*, 1993, 88(423): 881-889. DOI: 10.1080/01621459.1993.10476353.

[20] Martino L, Read J, Luengo D. Independent doubly adaptive rejection Metropolis sampling within Gibbs sampling. *IEEE Transactions on Signal Processing*, 2015, 63(12): 3123-3138. DOI: 10.1109/TSP.2015.2420537.

[21] Murphy K. An introduction to graphical models. Technical Report, University of California, 2001. https://www.cs.ubc.ca/~murphyk/Papers/intro_gm.pdf, March 2020.

[22] Friedman N, Geiger D, Goldszmidt M. Bayesian network classifiers. *Machine Learning*, 1997, 29(2/3): 131-163. DOI: 10.1023/A:1007465528199.

[23] Bilmes J A. A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models. Technique Report, International Computer Science Institute, 1998. http://lasa.epfl.ch/teaching/lectures/ML_Phd/Notes/GP-GMM.pdf, March 2020.

[24] Zivkovic Z. Improved adaptive Gaussian mixture model for background subtraction. In *Proc. the 17th International Conference on Pattern Recognition*, August 2004, pp.28-31. DOI: 10.1109/ICPR.2004.1333992.

[25] Murphy K P. Machine Learning: A Probabilistic Perspective. MIT Press, 2012.

[26] Pearson K. On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling. In *Breakthroughs in Statistics: Methodology and Distribution*, Kotz S, Johnson N L (eds.), Springer, 1992, pp.11-28. DOI: 10.1007/978-1-4612-4380-9_2.

[27] Balakrishnan N, Voinov V, Nikulin M S. Chi-Squared Goodness of Fit Tests with Applications. Academic Press, 2013.

[28] Das A, Kempe D. Approximate submodularity and its applications: Subset selection, sparse approximation and dictionary selection. *The Journal of Machine Learning Research*, 2018, 19(1): Article No. 3.

[29] Qian C, Yu Y, Zhou Z H. Subset selection by Pareto optimization. In *Proc. the Annual Conference on Neural Information Processing Systems*, December 2015, pp.1774-1782.

[30] Qian C, Shi J C, Yu Y *et al.* Parallel Pareto optimization for subset selection. In *Proc. the 25th International Joint Conference on Artificial Intelligence*, July 2016, pp.1939-1945.

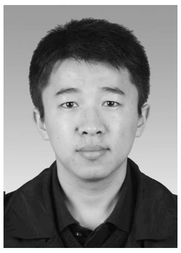
[31] Darrell W. A Genetic Algorithm Tutorial. *Statistics & Computing*, 1994, 4(2): 65-85.

[32] Lauritzen S, Spiegelhalter D. Local computations with probabilities on graphical structures and their application on expert systems. *J. Royal Statistical Soc.*: *Series B* (*Methodological*), 1988, 50(2): 157-194. DOI: 10.1111/J.2517-6161.1988.TB01721.X.

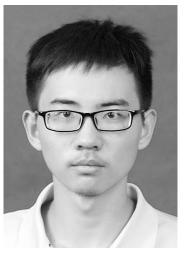
[33] Beinlich I, Suermondt H, Chavez R, Cooper G. The ALARM monitoring system: A case study with two probabilistic inference techniques for belief networks. In *Proc. the 2nd European Conf. Artificial Intelligence in Medicine*, August 1989, pp.247-256. DOI: 10.1007/978-3-642-93437-7_28.

[34] Oniśko A, Druzdzel M J, Wasyluk H. A probabilistic causal model for diagnosis of liver disorders. In *Proc. the 7th International Symposium on Intelligent Information Systems*, June 1998, pp.379-387.

[35] Conati C, Gertner A S, VanLehn K *et al.* On-line student modeling for coached problem solving using Bayesian networks. In *Proc. the 6th International Conference on User Modeling*, June 1997, pp.231-242. DOI: 10.1007/978-3-7091-2670-7_24.



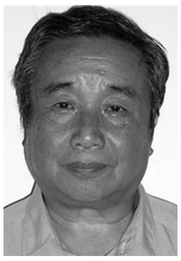
Jiao-Yun Yang received his B.S. degree and Ph.D. degree in computer science from University of Science and Technology of China, Hefei, in 2009 and 2014, respectively. He is an associate professor in the School of Computer Science and Information Engineering, Hefei University of Technology, Hefei. His research interests include heuristic search, machine learning and health computing. He is a member of CCF.



Jun-Da Wang is a undergraduate student in the School of Mathematics, Hefei University of Technology, Hefei. His research interests include Bayesian networks, computational learning theory and machine learning.

Yi-Fang Zhang is a undergraduate student in the School of Computer Science and Information Engineering, Hefei University of Technology, Hefei. Her research interest includes algorithm design and machine learning.

Wen-Juan Cheng received her B.S. degree in computer applications from Wuhan University, Wuhan, in 1993, and received her Master's degree and Ph.D. degree in computer applications from Hefei University of Technology, Hefei, in 2002 and 2012, respectively. She is a professor in the School of Computer Science and Information Engineering, Hefei University of Technology, Hefei. Her research interests include pattern recognition and computer application.



Lian Li received his B.S. degree in mathematics from Northwestern Normal University, Lanzhou, in 1976, and his Master's degree in mathematics from Lanzhou University, Lanzhou, in 1982. He is a professor in the School of Computer Science and Information Engineering, Hefei University of Technology, Hefei. His research interests include computational learning theory and machine learning. He is a member of CCF.