Research on the scope of capacity for different EDAs

Caichang Ding[1, 2, 3], Wenxiu Peng[2*]

[1]*State Key Lab of Software Engineering, Wuhan University, Wuhan 430072, China*

[2]*School of Computer, Wuhan University, Wuhan 430072, China*

[3]*School of Computer Science, Yangtze University, Jingzhou 434023, China*

Received 1 March 2014, www.cmnt.lv

Abstract

In this paper we investigation the scope of capacity for different EDAs to successfully solve problems, which concern to the mutual effects among the variables. More specifically, we study the learning restrictions that different EDAs confront to solve problems, which can be expressed by some ADFs. The research is conducted in the worst situation. The sub-functions in the ADFs are the same deceptive functions. We think that the capacity for this kind of algorithm are primarily influenced by the probabilistic model they depend on. We employ three different kind of EDAs so as to investigate the effect that the complexity of the probabilistic model has on the behave of the algorithm. Because the population size is crucial for EDAs, we use different population sizes in the experiments. Nevertheless, the results indicate that, in general, enlarge population size is not useful to solve more complex problems.

Keywords: EDAs, capacity, complexity, problem structure, probabilistic model

1 Introduction

Estimation of distribution algorithms (EDAs) [1-4] are a new kind of evolutionary algorithms (EAs) that use probabilistic models instead of the typical genetic operators used by genetic algorithms (GAs) [5, 6]. The features of the search space in EDAs are extracted by machine learning methods. In EDAs, employing a probabilistic model to represent the collected information which is used to generate new individuals later. By this way, probabilistic models can lead the search to hopeful areas of the search space.

Mathematically, an optimization problem can be seen as the minimization or maximization of a given function. Thus, optimization problems can be formulated as,

$$x^* = \arg\max_x f(x), \tag{1}$$

where $f: S \to \mathrm{R}$ is called the objective function or fitness function, $x = (x_1, \ldots, x_n) \in S$ is a candidate solution of the problem and S is named the problem space. In most cases, the optimum x^* is not unique. In this paper, the problem space S is an n dimensional discrete space.

Because the EDAs [2, 4] can capture the structure of the problem, EDAs are considered to be more efficient than GAs. In EDAs, the specific interactions among the variables of solutions in the problem are taken into mind. In evolutionary algorithms, the interactions are display implicitly in mind; whereas in EDAs, the interrelations are showed explicitly through the joint probability distribution associated with the individuals of variables selected from each generation. The probability distribution is calculated according to a population of selected candidate solutions of previous generation. Then, offsprings are sampled from this probability distribution generate. Neither crossover nor mutation has been used in EDAs. Figure 1 displays the Pseudo-code of Estimation of Distribution Algorithms.

1 $D_{t=0} \leftarrow$ gerate N individuals randomly
2 **Do**
3 $D_t \leftarrow$ evaluate individuals
4 $D_t^{Se} \leftarrow$ select $M < N$ individuals from D_t according to a selection method
5 $p_t(x) = p(x \mid D_t^{Se}) \leftarrow$ estimate the joint probability distribution by means of a probabilistic model
6 $D_{t+1} \leftarrow$ sample M individuals from $p_t(x)$ and creat the new population
7 $t = t + 1$
8 **until** stopping criterion is met

FIGURE 1 Pseudo-code of Estimation of Distribution Algorithms

An EDA has such basic elements [2]: encoding of candidate solutions, objective function, selection of parents, building of a structure, generation of offspring, selection mechanism, and algorithm parameters like population size, selection size, etc.

Different EDAs [2] mainly differ in the kind of probabilistic models employed and the approaches used to learn, then sample from the obtained models. Bayesian

[*]***Corresponding author*** e-mail: hamigua_ping@hotmail.com

networks is one of the models that has been widely used in EDAs. One of the advantages of EDAs that employ these kinds of models is that the complexity of the learned structure relies on the characteristics of the selected individuals. Moreover, the analysis of the models learned during the search can provide useful information about the problem structure.

How the characteristic of the search space are reflected in the learned probabilistic models is another related and important issue in EDAs. This issue has received special attention from the EDA community, and is essential to understand the mechanisms that enable EDAs to efficiently sample from the problem space during the search process. However, the question of analysing the relationship between the problem space and the structure of the learned probabilistic models becomes more and more difficult because of the next two main reasons: the search process is random in the EDAs, and the methods used when learning the models can only detect approximate, suboptimal, structures.

To investigate the effect that the complexity of the probabilistic model has on the behave of the algorithm, we employ three different implementations. Firstly, an EDA which is called univariate marginal distribution algorithm (UMDA) [7] that is considered independent among the variables, secondly, an EDA which is called EDAdt [8] uses a dependency-tree model every generation and last, an EDA which is called estimation of Bayesian networks algorithm (EBNA) [9,10] uses Bayesian networks. Because the population size is crucial [11] for EDAs and particularly when the used probabilistic model is Bayesian networks, we employ varying population sizes.

The experimental results indicate that the ability of EDAs to solve the testing functions is lost quickly when the number of sub-functions exceeds a certain value in the objective function. This threshold of behave indicates a obvious phase-transition phenomenon that clearly delimits the frontiers of effectiveness. The experimental results also indicate that the ability to learn structures is very important to extend the restrictions of successful EDA implementations. However, EBNA displays a tremendous decreasing of performance because the learning method cannot build more sophisticated models. Moreover, according to the experimental results, the learning of unrestricted Bayesian networks requires a huge computational cost for solving the problems. Therefore, in order to solve the problems, the complexity of the models should grow exponentially along with growth of the number of sub-functions.

2 Probabilistic models

The probabilistic model associates to the qualitative and quantitative structure determined by the probability function. Though the EDAs could employ any kind of probabilistic model, the widely used models are those that show their qualitative component by a graph. Especially, one kind of models that has been widely used in EDAs are Bayesian networks.

Bayesian networks [12-14], which is called belief networks are a kind of probabilistic graphical model. This kind of probabilistic models is one of very popular paradigms that can deal with probability distributions efficiently in modelling uncertain information. The domain of expert systems is one of the most important sources for the development of Bayesian networks. Moreover, in the past few years, Bayesian networks have obtained considerable attention in domain of the machine learning. As a result of this attention, more and more papers and tutorials have appeared. Thus, besides expert systems, Bayesian networks are also applied in classification problems, bioinformatics and optimization.

Bayesian networks are the product of associating probability and graph theory [15], similarly with any other probabilistic graphical model. The graphical consist of the model encodes a number of conditional independences related to the probability distribution. Let $X=(X_1,\ldots,X_n)$ be an n dimensional discrete random variable. A Bayesian network is a graphical expression of the factorization of the joint probability distribution for X, $p(X=x)$, where $x=(x_1,\ldots x_n)$ is an assignment of the random variable X. More specifically, a Bayesian network can be expressed as a pair (s,θ_s)s where s is a directed acyclic graph that is model structure and θ_s is the set of parameters associated to the structure that is model parameters. The structure s determines the set of conditional dependences among the random variables of X. According to the structure s, the joint probability distribution $p(x)$ can be factorized by means of marginal and conditional probability functions. More specifically, the probability distribution factorizes according to the graph as,

$$p(x)=\prod_{i=1}^{n} p(x_i \mid pa_i), \tag{2}$$

where pa_i denotes a value of the variables Pa_i that is the parent set of X_i in the graph s.

The local probability distributions in the factorization are those which is induced by means of the product that appears in Equation (2). We suppose that these local probability distributions depend on the parameters $\theta_s=(\theta_1,\ldots,\theta_n)$. So Equation (2) could be rewritten by specifying the parameters:

$$p(x \mid \theta_s)=\prod_{i=1}^{n} p(x_i \mid pa_i,\theta_i) \tag{3}$$

Suppose that the variable X_i has r_i possible values, thus, the local probability distribution $p(x_i \mid pa_i^j,\theta_i)$ is an unbounded discrete distribution:

$$p(x_i^k \mid pa_i^j,\theta_i)=\theta_{ijk}, \tag{4}$$

where $pa_i^1,\ldots,pa_i^{q_i}$ represent the q_i possible values of the parent set Pa_i. The parameter θ_{ijk} denotes the probability of variable X_i which takes in its *k*-th value, at the same time, the set of its parents' variables takes in its *j*-th value. Therefore, the local parameters are determined by $\theta_i = ((\theta_{ijk})_{k=1}^{r_i})_{j=1}^{q_i}$.

In order to look for a Bayesian network [16, 17], which can make us to represent and deal with the uncertain knowledge of a specific field, setting both the structure and the parameters is very necessary. The structure and conditional probabilities that is necessary for describing the Bayesian network can be provided either externally by experts, by machine learning from datasets or by mixing both of these methods. In this paper. We mainly focus on the second method. Besides, when the structure has been automatically learned, it can provide us with perceptions into the interactions between the variables of the field.

The learning step can be separated into two subtasks that are structural learning and parameter learning [18-20]. Although there are different approaches to learn the structure of a Bayesian network, we mainly focus on the so-called score plus search method. This kind of methods copes with the structure learning as an optimization problem. Thus, the steps of learning a Bayesian network can be expressed as follows. Given a data set D containing N cases, $D=\{x_1,\ldots,x_N\}$, finding the structure s^* such that,

$$s^* = \arg\max_{s\in S^n} g(s,D), \tag{5}$$

where $g(s,D)$ is the score which measures the quality of any given structure s related to the data set D, and S^n is the set of all possible directed acyclic graphs (DAG) which have *n* nodes. A number of relevant and used heuristic techniques such as greedy search, simulated annealing, particle swarm optimization, genetic algorithms and ant colony optimization have been used in this task.

If score can be decomposed in presence of complete data sets, it is the one of the desirable character. These scores can be decomposed in sub-scores related to every node X_i and its parents Pa_i in the structure s. Formally, we can express a decomposable score as:

$$g(s,D) = \sum_{i=1}^{n} g_D(X_i, Pa_i), \tag{6}$$

where the g_D is the sub-score function. As a result of the decomposability, it is computationally more efficient when the local search is carried out, because when we add an arc into the network, it is only necessary to evaluate the set of nodes involved by this change.

When we have defined a score to assess Bayesian networks, we have to run a search process to look for the Bayesian network, which can return the best score given the data.

In practical applications, we must to look for an suitable model structure [21-25] as soon as possible. Thus, a simple learning method, which can find a relatively good structure, even though it is not best, is preferred. There are a number of learning methods, which can be employed for this task. However, a specific search algorithm which is called Algorithm B [19], which is typically used by most of Bayesian network based EDAs.

Algorithm B is a greedy search algorithm and its pseudo-code is presented in Figure 2, where D is a data structure which contains the information needed to deal with the candidate arcs which should be added into the network. At the beginning, Algorithm B starts from an arc-less structure, which represents independently among the variables, and at each iteration, an arc is added into network that can increase the score greatly. The algorithm stops when no arc that can increase the score any more, can be added into the network.

1 begin with an arcless strcture
2 compute $D[X_j \to X_i] = g_D(X_i, X_j) - g_D(X_i)$ for all distic X_i, X_j
3 **do**
4 find the largest $D[X_j \to X_i]$, and add an arc from X_j to X_i in the structure
5 $D[X_j \to X_i] = g_D(X_i, Pa_i \cup X_j) - g_D(X_i)$ for all disctinct X_i, X_j not belonging to Pa_i
6 $D[X_j \to X_i] = -\infty$
7 **until** every $D[X_j \to X_i] < 0$

FIGURE 2 Pseudo-code for Algorithm B

When we have got a Bayesian network by learning, this model could provide us with detailed probabilistic information which we are interested in. Usually, the information, which the researcher wants to know is the probability of some events on the basic of special observations. Generally speaking, the probabilities which we are interested in, are not reposted in the Bayesian network obviously. It is necessary to compute in order to obtain them,. This course is called probabilistic inference and it is usually an NP-complete problem.

Emulation of Bayesian networks, which is also named stochastic sampling, can be regarded as an option to the exact inference. The Emulation of a given probabilistic graphical model requires to get a sample from the probability distribution for X which the model encodes. Next, the marginal and conditional probabilities involved can be calculated from the sample.

For our goals about EDAs, the intension of the emulation of Bayesian networks is to get a new population in which the probability relations among the random variables of the network are potential. Specifically, for the purpose of sampling from the Bayesian network, the sampling method which we employ is forward. The variable must be sampled after all its parent variables have

been obtained. This approach is named probabilistic logic sampling (PLS). Figure 3 presents a pseudo-code of this approach.

```
1   π ← ancestral ordering of the nodes in the baysian network
2   for j = 1 to N
3     for i = 1 to n
4       x_{j,π(i)} ← randomly generate a value form p(x_{π(i)} | pa_{π(i)})
5     done
6   done
```

FIGURE 3 Pseudo-code of the probabilistic logic sampling method

3 Testing functions

In order to study the performance of EDAs when the complexity of the function grows, we handle additively decomposable functions (ADFs). This kind of functions are regularly employed by researchers [26-28]. As everyone knows, a number of optimization problems investigated recent years could be modelled by ADFs. The model of function employed in this paper, in which new sub-functions are one by one added, could be seen as a system that grows its complexity with the time, because new interactions appear among the variables [29].

We define ADFs in a general form. Let $S=\{0,1\}^n$ be the problem space, a fitness function $f:S\to R$ is decomposable if it can be expressed as a sum of sub-functions of lower dimension,

$$f(x)=\sum_{c_i\in C} f_i(c_i), \tag{7}$$

where $x=(x_1,\ldots,x_n)\in\{0,1\}^n$ and $C=\{c_1,\ldots,c_l\}$ is a set composed of distinct sub-sets $c_i\subseteq\{x_1,\ldots,x_n\}$. Moreover, we suppose that set c_i and c_j are distinct and not include each other. This kind of functions is also featured by its order *k*, which is determined by the size of the largest subset in C.

In this paper we employ detailed cases of this general kind of functions. Firstly, all the sub-sets in C have three variables, that is to say $k=3$. Thus, C includes all the sets of distinct sub-sets, which are selected from all the $C(n,k)$ possible sub-sets of variables. Secondly, all the sub-functions f_i are the same deceptive function f_{3dec} which is proposed by Goldberg. Given $u(y)=\sum_{i=1}^{k} y_i$, where $y\in\{0,1\}^k$, the function can be formulated as,

$$f_{3dec}(c_i)=\begin{cases}0.9 & for \quad u(c_i)=0\\ 0.8 & for \quad u(c_i)=1\\ 0.0 & for \quad u(c_i)=2\\ 1.0 & for \quad u(c_i)=3\end{cases}. \tag{8}$$

From standpoint of the EDA, the advantages of the fitness functions that we use are evident [30-32]. First of all, independently of the number of sub-functions, we all time know the globally optimal solution, which is with all ones. Secondly, the deceptive method generates strong interactions among the variables which belong to every sub-function.

About the functions f_{3dec}, when they are managed without overlapping among the set of variables, we get the function Deceptive3. This function was designed in the background of genetic algorithms with the purpose of analysing their restrictions. Thus, in this paper, the function will be a helpful reference so as to analyse the restrictions of behave in EDAs. At present, deceptive separable functions are widely employed to analyse evolutionary algorithms.

For the sake of raising the complexity of the functions step by step, we use a simple method. First of all, we create a series of objective functions in which every new function puts one more sub-function into the previous one. This series of functions is determined by the ordered set $C=\{c_1,\ldots,c_l\}$. This ordered set is composed of l different sub-sets of variables which are randomly taken from all the $C(n,k)$ possible combinations following a uniform distribution. Though we could add all the $C(n,k)$ different subsets in C, it will be not essential to get learning restrictions. Therefore, the *s*-th objective function in the list adds *s* sub-functions which are the first *s* sub-sets of variables in C. The *s*-th function can be formulated as,

$$f_s=\sum_{i=1}^{s\le l} f_{3dec}(c_i). \tag{9}$$

However, the ordered set C has a constraint. The join of the first n/k sub-sets equal to the whole set of n variables, constituting the function Deceptive3. Be aware that in the functions from $s=1$ to $s=n/k$, a number of the variables do not appear in any sub-function. In order to make these functions intact, we directly use the previously mentioned function u to include the set of variables $\{x\setminus\bigcup_{i=1}^{s} c_i\}$, which do not appear in the sub-function. This measurement is very helpful to investigate the univariate and bivariate EDA. Moreover, this separable function is helpful to indicate difficulty of a problem.

Finally, because of the random nature of the set C, we have produced for the experiments 100 different random instances of this kind of sets and the results presented are the average value from them. The entire set of experiments include three different problem sizes ($n\in\{24,48,72\}$) and the maximum number of sub-functions which is

determined by C, is $l = 200$. Moreover, for every possible function, we conduct ten independent EDA runs. The number of runs per instance is limited because of the high computational cost.

The intensity of interaction can be understood as a notion about the interdependences that appear among the variables of a problem. Though there are a number of different methods to estimate this notion, in the background of the present work, we suppose that the intensity of interaction is simply determined by the number of sub-functions contained in the objective function [33-35].

Moreover, in order to give a more direct measure of the intensity of interaction, we consider the frequency that every variable appears in sub-functions. For instance, in some ADFs, each variable appears in only one sub-function. Generally speaking, given s sub-functions with size k including n variables, we compute the expected number of variable appearing in sub-functions. It is formulated as,

$$\langle s \rangle = s\frac{k}{n}. \tag{10}$$

In order to demonstrate how the landscapes of the ADFs change along with the number of sub-functions increasing, we give a simple example in which the number of variables is 9 in Figures 4-7.

These figures shows changes of function value with four objective functions, which differ only in the number of variables. The solutions are sorted by the number of ones so as to supplying more intuitive graphs. For instance, the region with the number 4 is comprised of all the $C(9,4)$ solutions, which have 4 ones. Therefore, the region with the number 0 and number 9 only contains one solution. The two solutions play an important role in the kind of functions we employ, and thus, the solution with all zeros is emphasized with a circle and the solution with all ones, which is optimum is emphasized with two concentric circles.

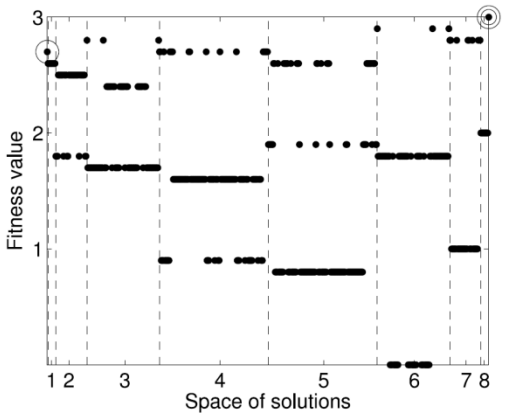

FIGURE 4 f_3, 3 sub-functions, $\langle s \rangle = 1$

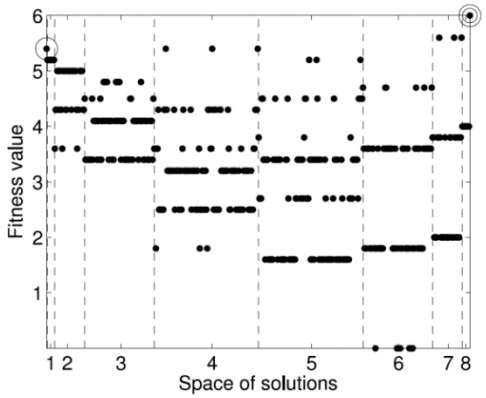

FIGURE 5 f_6, 6 sub-functions, $\langle s \rangle = 2$

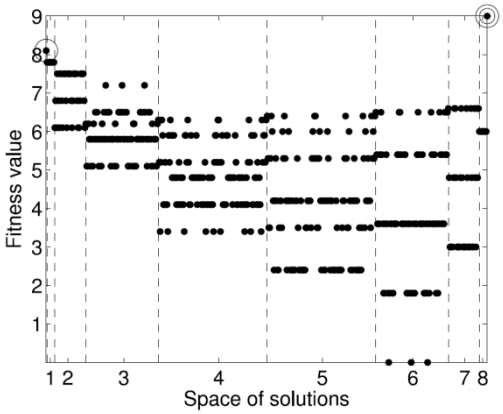

FIGURE 6 f_9, 9 sub-functions, $\langle s \rangle = 3$

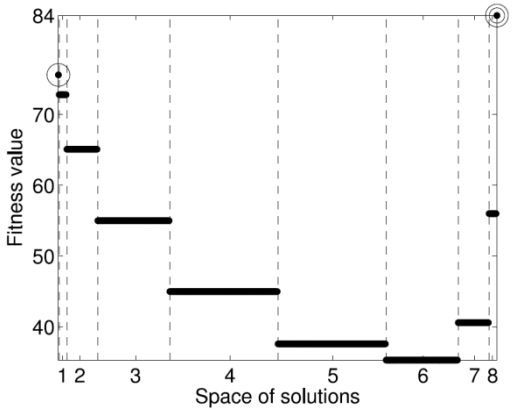

FIGURE 7 f_{84}, 84 sub-functions, $\langle s \rangle = 28$

In this part, for each function created, we compute the order of a possible related exact factorization. This computational process is a proximity to the exact factorization, which has minimum complexity. However, this process provides useful information about the complexity of the probabilistic models, and it is essential for an EDA to solve the created problems in the worst conditions. The orders of the used functions change as shown in Figure 8. This Figure presents when problem sizes is 3, the order change with the average number of sub-functions in which each variable appear. We begin from $\langle s \rangle = 1$, in this case it is the Deceptive3 function. The results show an exponential growth of the number of parameters related to exact factorizations and thus, the

complexity of these models rapidly becomes too high to manage. Even if having enough knowledge concerning the function, looking for this kind of probabilistic models is confronted with considerable computational restrictions. Moreover, if these factorizations were employed in an EDA, the population size should also grow with the complexity of the factorizations so as to get a robust behave of EDAs [36].

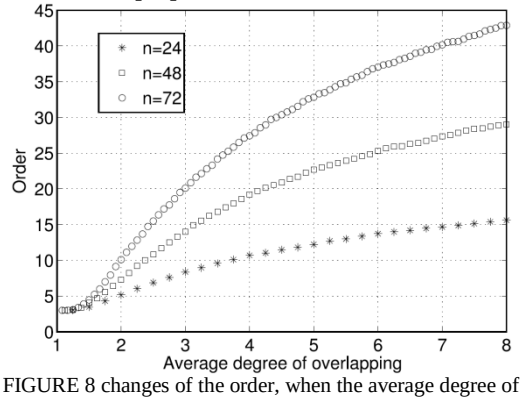

FIGURE 8 changes of the order, when the average degree of overlapping grows

4 Experimental results

In this part, we present experimental results. The relationship between the ratio of success and the number of sub-functions for each kind of algorithm gives in Figure 9, Figure 11 and Figure 13. The relationship between hamming distance to the best solution and the number of sub-functions for each kind of algorithm gives in Figure 10, Figure 12 and Figure 14. We only show the experimental results for the problems with $n = 72$ variables. The performance of the algorithms is alike for the three different problem sizes ($n \in \{24, 48, 72\}$) that we have used. Nevertheless, when the number of variables grows, the patterns are more obvious and clearer.

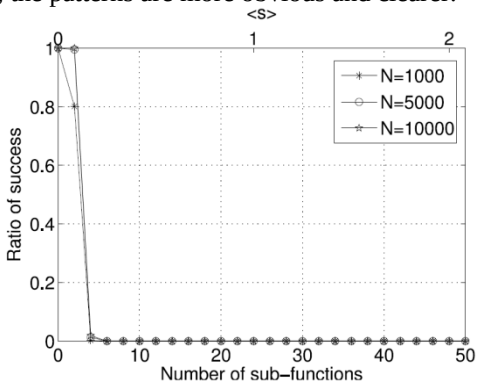

FIGURE 9 Ratio of success in UMDA

Generally speaking, by means of the descriptors employed in Figure 9 - Figure 14, we can obviously see the behave that different EDA implementations fails. The graphs exhibit a phase-transition effect when the interaction in the created problems exceeds a certain degree. However, this effect is especially remarkable in the graphs of ratio of successful runs, which fall off from 1 to 0 steeply after only adding very few sub-functions. Therefore, in UMDA and EDAdt, the number of sub-functions between total success and complete failure is within 2 sub-functions. For the EDA based Bayesian networks, though this algorithm also suffers a sudden fail, the conversion from 1 to 0 in the ratio of successful runs is relatively more gradual. concerning the hamming distance, it exhibits a more gradual change, which supplies complementary information concerning the quality of the solutions. For instance, though in Figure 9 the ratio of success can be equal to 0 when the number of sub-functions is 4, UMDA returns solutions which are near the optimum in hamming distance in Figure 10.

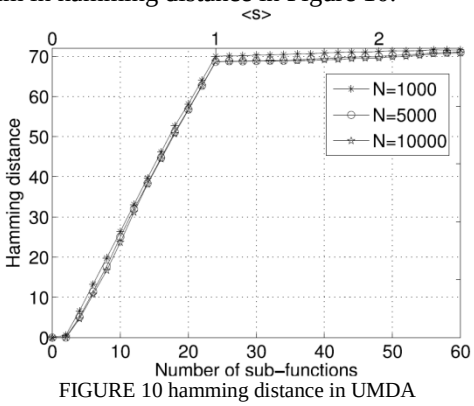

FIGURE 10 hamming distance in UMDA

From the results obtained from the experiment, we also achieve the understanding of the effects that the probabilistic models and the population size have on the EDAs when we solve problems with growing interaction. As anticipated, the probabilistic model employed in the EDAs has a determined influence on the scope of problems that it can solve. Therefore, UMDA begins to fail when the number of sub-functions is 2 in the objective functions, EDAdt can attain the equivalent level of Deceptive3 function and EBNA collapses between the ADFs and the functions with $\langle s \rangle = 2$.

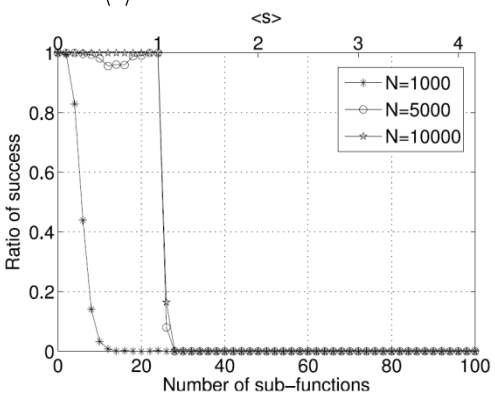

FIGURE 11 ratio of success in EDAdt

In addition, according to experimental results, if an EDA to can deal with more complex structural models, the population size has a more effect on the performance of the EDA. It is decisive to achieve a robust behaviour of EDAdt and EBNA, while UMDA is hardly affected by this parameter. As presented in Figure 11 and Figure 12, the smallest population size ($N = 1000$) is clearly inadequate,

therefore these algorithms cannot achieve a ideal performance.

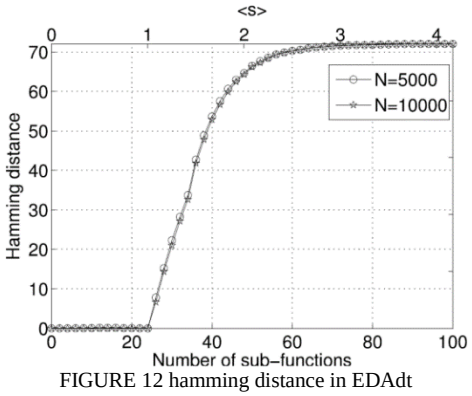

FIGURE 12 hamming distance in EDAdt

The greatest influence of the population size takes place in EBNA and it is reflected in the graphs shown in Figure 13 and Figure 14. However, even for EDA based Bayesian networks, this parameter exhibits a limited effect to overcome influence of threshold. This suggests that, though the population size is important to obtain a ideal behavior, growing this parameter is not an efficient way to solve the problems when the degree of interaction grows. In this respect, we can see in Figure 13 and Figure 14 that the different graphs tend to be closer as the size of the population grows.

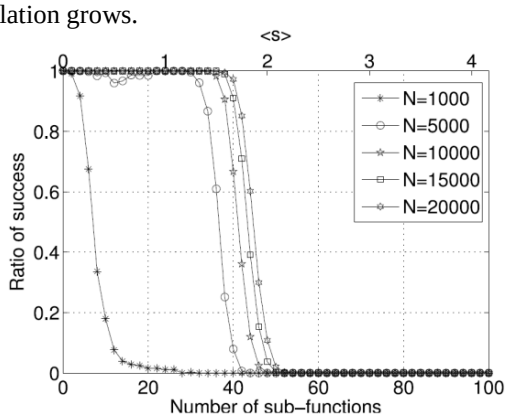

FIGURE 13 ratio of success in EBNA

From the graphs of hamming distance, we can see another phenomenon. when the interaction in the problem exceeds a certain extent, all EDAs only can find solution with all zeros. Thus, we could conclude that, from the aspect of efficiency, UMDA is the best choice to deal with the problems which exceed this critical threshold of difficulty. According to this viewpoint, and considering the whole scope of functions that can be created from $s=0$ to $s=C(n,k)$, EBNA exhibits better performance in a reduced sub-space. Be aware that Figure 9 - Figure 14 only presents the behaviour of the EDAs at the first ranks of difficulty. When n equal to 72 and *k* equal to 3, we could obtain in the fitness function up to $C(72,3)=59640$ sub-functions. Thus, the graphs only depict a small fraction of this number.

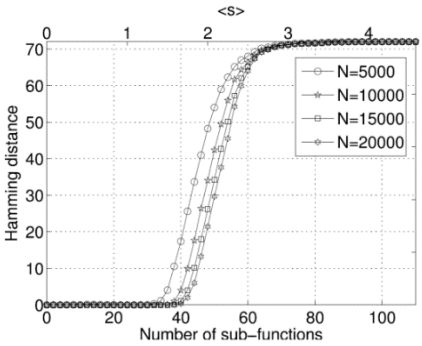

FIGURE 14 hamming distance in EBNA

From the experimental results given in Figure 9- Figure 14, we have observed that how the behave of different EDAs suddenly fails when the interaction among the problem variables exceeds a certain extent. The reason for this phenomenon is that UMDA and EDAdt are deficient in ability to learn models. Nevertheless, for EBNA, it is deserving to carry out a more thorough analysis of the reason of its failure. In order to do this, we consider the complexity of the Bayesian models learned from the experiment. We employ the order of the factorizations which is determined by these probabilistic models to assess their complexity. In Figure 15, we present the average maximum orders of the models learned from each execution.

Be aware that the influence of phase transition seen in Figure 13 and Figure 14 is associated with Figure 15. Therefore, EBNA starts to collapse quickly after the peaks in Figure 15, that is to say when the algorithm could not construct more complex models. Figure 15 indicates that the complexity of the Bayesian networks grows exponentially with the number of sub-functions so as to solve the problems. Be aware that when the learning method cannot construct the appropriate structures to solve more problems, the algorithm still spends a large number of computational resources on learning false models.

It is a question that what extent the behaviour mentioned above depends on the learning method, which need an thorough analysis. We argue that, in this worst situation, we will likely get a resembling performance for other approximate learning methods.

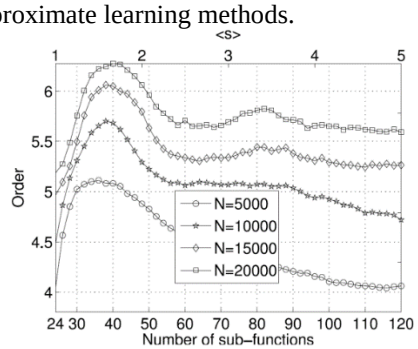

FIGURE 15 maximum orders of the models learned by EBNA

We compare the complexity of the learned structures with the complexity of the exact factorizations in Figure 16 and Figure 17. The curve shown in Figure 16 corresponds to the order of the exact factorizations, which were presented in Figure 8. However, in the graph, the number of

variables is only for $n = 72$ and curve is exhibited in relation to the number of sub-functions. We have used a circular mark so as to approximately indicate the region where EBNA fails. In Figure 17, we put the order of the exact factorizations and the order of the Bayesian networks obtained by the EDA together. The dashed line indicates the graph of the exact factorizations. In this Figure, we can observe that, when the EDA can run successfully, the maximum complexity of the models obtained is not less than the complexity of the exact factorizations. That is to say, when the EDA cannot learn the complexity of the models, which equals to the exact factorizations, it no longer solves more complex problems. The dashed line of the factorizations exactly separates the problems solved by EDA from the problems unsolved by EDA for every population size. Population size, which is insufficient may be the main reason to explain the bad behave in looking for the model structure.

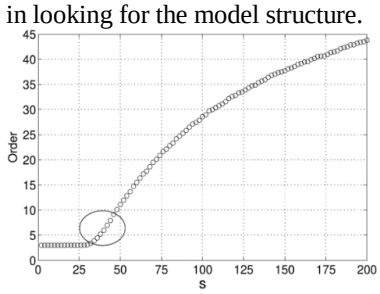

FIGURE 16 maximum orders of the models learned by EBNA

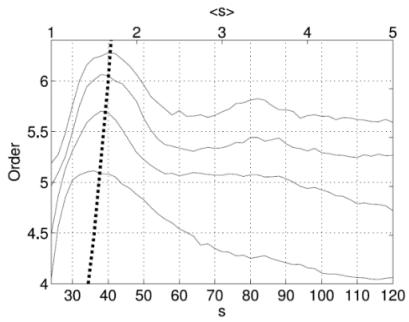

FIGURE 17 maximum orders of the models learned by EBNA

5 Conclusions

In this paper we have investigated the restrictions of behave that different EDA implementations confront when the interaction among the variables of the problem grows. We conduct the study by the employment of ADFs in which new sub-functions are added one by one. Therefore, the interaction can be measured by the number of sub-functions, which the objective function contains. In addition, we employ the separable deceptive function as a indication of problem difficulty so as to supply more perspicuous results. In the experiments, we have carried out three different type of EDA implementations. Because these algorithms only distinguish in the probabilistic model employed, the results indicate the influence that using more complex models has so as to solve a broader scope of problems. We have also employed different population sizes, which has been crucial so as to realize a potent performance in EDAs that base on Bayesian networks. Nevertheless, the results show that, generally speaking, growing this parameter is not useful to solve more difficult problems.

We have found that the EDAs fail with a phenomenon of phase transition when the number of sub-functions in the objective function grows in the worst situation,. The area in which EBNA collapses is between the separable deceptive functions and the objective functions with $2n/k$ sub-functions. The reason for the breakdown of the algorithm is that the EDA cannot obtain correct models. Once the EDA cannot to learn more complex models which is essential to solve more difficult problems, the algorithm fails. The complexity of the networks tends to grow exponentially so as to find the optimum needed for the algorithm. Nevertheless, exceeding a certain degree, the learning method cannot structure the suitable models to solve the problem and then, the algorithm collapses quickly. It implies that, when the degree of interaction exceeds a critical point, the learning of Bayesian networks may not be able to obtain the information needed to find the optimum from the population. Moreover, the relationship between the models learned by the EDA and the exact factorization indicates forceful computational restrictions because of the exponential increasing complexity of the structural essential to solve the problems.

The restrictions of effectiveness displayed in this paper are straight associated to the learning procedure of the algorithm. Nevertheless, these restrictions do not always have only one cause. We can confirm three different standpoints from which the learning restrictions in EDAs could be investigated:

1) restrictions of the learning either because the model needs a priori knowledge or because they learn approximate models with bounded complexity.

2) Even if we employ a desired learning method, there are efficiency restrictions because the complexity of models grows exponentially, which is disadvantage to solve the problems when the number of interactions grows.

3) Restrictions for the population either because of the deficiency of information that it includes to solve the problem or because this parameter should grow exponentially to supply an effective learning.

In general, we have investigated the concept of borders of EDA effectiveness concerning the extent of interaction of the problem. The main objective of this study is not to look for the best algorithm or negate any method but to comprehend which algorithms are the suitable for which problems.

Acknowledgments

This paper is supported in part by National Natural Science Foundation of China (Grant no.60975050), Research Fund for the Doctoral Program of Higher Education (Grant no.20070486081) and Fundamental Research Funds for the Central Universities (Grant no.6081014). The authors are grateful to the anonymous referees for their insightful and constructive comments, which greatly improved the quality of the paper.

References

[1] Mühlenbein H, Paaß G 1996 From recombination of genes to the estimation of distributions I. Binary parameters Lectures Notes in Computer Science: Parallel Problem Solving from Nature (PPSN IV) **1141** *Springer Verlag* 178-87

[2] Larrañaga P, Lozano J A 2002 Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation *Kluwer Academic Publishers*

[3] Armañanzas R, Inza I, Santana R, Saeys Y and Larrañaga P 2008 A review of estimation of distribution algorithms in bioinformatics BioData Mining **1**(6) 1–12

[4] Höhfeld M, Rudolph G 1997 Towards a theory of population based incremental learning *Proceedings of the 4th International Conference on Evolutionary Computation IEEE Press*

[5] Goldberg D E 1989 Genetic Algorithms in Search, Optimization, and Machine Learning *Addison: Wesley*

[6] Eiben E A and Smith J E 2003 Introduction to Evolutionary Computing (Natural Computing Series) *Springer*.

[7] Mühlenbein H 1998 The equation for response to selection and its use for prediction Evolutionary Computation **5**(3) 303–46

[8] Baluja S, Davies S 1997 Using optimal dependency-trees for combinatorial optimization: Learning the structure of the search space *Proceedings of the 14th International Conference on Machine Learning Morgan Kaufmann* 30-8

[9] Etxeberria R and Larrañaga P 1999 Global optimization using Bayesian networks *Proceedings of the Second Symposium on Artificial Intelligence Habana* 151-73

[10] Santana R, Larrañaga P, Lozano J A 2005 Interactions and dependencies in estimation of distribution algorithms *Proceedings of the 2005 Congress on Evolutionary Computation IEEE Press* 1418–25

[11] Wu H, Shapiro J L 2006 Does over-fitting affect performance in estimation of distribution algorithms *Proceedings of the 8th annual conference on Genetic and evolutionary computation ACM Press* 433–4

[12] Meek C 1995 Strong completeness and faithfulness in Bayesian networks *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann Publishers* 411-8

[13] Neapolitan R E 2003 Learning Bayesian Networks *Upper Saddle River: Prentice Hall*

[14] Jensen F V, Nielsen T D 2007 Bayesian Networks and Decision Graphs *Springer*

[15] Koller D, Friedman N 2009 Probabilistic Graphical Models: Principles and Techniques *Cambridge: MIT Press*

[16] Friedman N, Geiger D, Goldszmidt M 1997 Bayesian network classifiers *Machine Learning* **29**(2) 131–63

[17] Hauschild M, Pelikan M, Sastry K, Goldberg D E 2012 Using previous models to bias structural learning in the hierarchical *BOA Evolutionary Computation* **20(**1) 135–60

[18] Friedman N, Linial M, Nachman I. 2000 Using Bayesian networks to analyze expression data *Journal of Computational Biology* 7 601–20

[19] Buntine W 1991 Theory refinement on Bayesian networks *Proceedings of the Seventh Conference on Uncertainty in Artificial Intelligence San Mateo Morgan Kaufmann* 52-60

[20] Chickering D M, Geiger D, Heckerman D 1995 Learning Bayesian networks: Search methods and experimental results *Proceedings of the Fifth International Workshop on Artificial Intelligence and Statistics* 112-28

[21] Roberto S 2005 Estimation of distribution algorithms with Kikuchi approximations *Evolutionary Computation* **13**(1) 67-97

[22] Ocenasek J 2006 Entropy-based convergence measurement in discrete estimation of distribution algorithms Towards a New Evolutionary Computation: Advances on Estimation of Distribution Algorithms *Springer* 39–50

[23] Pelikan M, Sastry K and Goldberg D E 2005 Multi-objective hBOA, clustering, and scalability *Proceedings of Conference on Genetic and Evolutionary Computation, ACM Press* 663-70

[24] IInza P, Larrañaga P, Etxeberria B S 2000 Feature subset selection by Bayesian network-based optimization *Artificial Intelligence* **123**(1) 157-84

[25] Saeys Y, Degroeve S, Aeyels D, van de Peer Y and Rouze P 2003 Fast feature selection using a simple estimation of distribution algorithm: a case study on splice site prediction *Bioinformatics* **19**(s2) 179-88

[26] Joaquin R and Roberto S 1999 Improving the discovery component of classier systems by the application of estimation of distribution algorithms *Proceedings of the Students Sessions*

[27] Santana R 2002 An analysis of the performance of the mixture of trees factorized distribution algorithm when priors and adaptive learning are used *Institute of Cybernetics, Mathematics and Physics*

[28] Coffin D J and Smith R E 2007 The limitations of distribution sampling for linkage learning *Proceedings of the 2007 Congress on Evolutionary Computation IEEE Press* 364–9

[29] Naudts B and Kallel L 2000 *IEEE Transactions on Evolutionary Computation* **4**(1) 1–15

[30] Li X, Mabu S and Hirasawa K 2014 *IEEE transactions on evolutionary computation* **18**(1) 98-113

[31] Robles V, Peña J M, Pérez M S and Herves V 2006 GA-EDA: A new hybrid cooperative search evolutionary algorithm Towards a New Evolutionary Computation: Advances in Estimation of Distribution Algorithms *Springer* 187–220

[32] Armañanzas R, Saeys Y, Inza I, Garca-Torres M, Bielza C, van de Peer Y and Larrañaga P 2011 Peakbin selection in mass spectrometry data using a consensus approach with estimation of distribution algorithms *IEEE/ACM Transactions on Computational Biology and Bioinformatics* **8**(3) 760–74

[33] Hauschild M, Pelikan M, Sastry K, Lima C 2009 *IEEE Transactions on Evolutionary Computation* **13**(6) 1199–217

[34] Abdollahzadeh A, Reynolds A, Christie M 2012 Bayesian optimization algorithm applied to uncertainty quantification *SPE Journal* **17**(03), 865-73

[35] Soto M R, González-Fernández Y and Ochoa A 2012 Vine estimation of distribution algorithm *Proceedings of the VIII Congreso Español sobre Metaheurísticas Algoritmos Evolutivos y Bioinspirados*

[36] Mühlenbein H 2012 Convergence of Estimation of Distribution Algorithms Markov Networks in Evolutionary Computation *Springer* 91–108

Authors

Caichang Ding, born on October 10, 1980, Hubei, China

Current position, grades: Ph.D. student at State Key Lab of Software Engineering, Wuhan University, Wuhan, China.
University studies: M.Sc. degree from the School of Computer, Wuhan University, Wuhan, China, in 2006.
Scientific interest: computational learning theory, statistical learning, basic theory of evolutionary computation and optimization theory.
Publications: 5 papers.
Experience: Caichang Ding received the B.Sc. degree from the School of Mechanical & Electronic Information, China University of Geosciences, Wuhan, China, in 2003, and the M.Sc. degree from the School of Computer, Wuhan University, Wuhan, China, in 2006. He is currently a Ph.D. student at State Key Lab of Software Engineering, Wuhan University, Wuhan, China, and a lecturer in the School of Computer Science, Yangtze University, Jingzhou, China.

Wenxiu Peng, born on February 14, 1981, Hubei, China

Current position, grades: lecturer at School of Computer Science, Yangtze University, Jingzhou, China.
University studies: M.Sc. degrees from Hubei University, Wuhan, China in 2006.
Scientific interest: computational learning theory, statistical learning, basic theory of evolutionary computation and optimization theory.
Publications: 3.
Experience: Wenxiu Peng received the B.Sc. and M.Sc. degrees from Hubei University, Wuhan, China, in 2003 and 2006. She is currently a lecturer in the School of Computer Science, Yangtze University, Jingzhou, China. Her main research interests include computational learning theory, statistical learning, basic theory of evolutionary computation and optimization theory.