Applied Soft Computing 11 (2011) 2974–2989

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Subspace estimation of distribution algorithms: To perturb part of all variables in estimation of distribution algorithms

Helong Li[a,b,*], Yi Hong[b], Sam Kwong[b]

[a] Department of Electronic Commerce, South China University of Technology, Guangzhou 510006, China
[b] Department of Computer Science, City University of Hong Kong, Kowloon, Hong Kong

ARTICLE INFO

Article history:
Received 13 October 2009
Received in revised form 31 May 2010
Accepted 28 November 2010
Available online 9 December 2010

Keywords:
Estimation of distribution algorithms (EDAs)
Subspace estimation of distribution algorithms (subEDAs)
Univariate marginal distribution algorithm (UMDA)
Estimation of Bayesian network algorithm (EBNA)

ABSTRACT

In the traditional estimation of distribution algorithms (EDAs), all the variables of candidate individuals are perturbed through sampling from a probability distribution of promising individuals. However, it may be unnecessary for the EDAs to perturb all variables of candidate individuals at each generation. This is because one variable may be dependent on another variable and all variables may have different saliences even if they are independent. Therefore, only a subset of all variables in EDAs really function at each generation.

This paper proposes a novel class of EDAs, termed as subspace estimation of distribution algorithms (subEDAs), from a new perspective to reduce the space of variables for use in model building and model sampling based on EDAs' performance. In subEDAs, only part of all variables of candidate individuals are perturbed at each generation. Three schemes are described in details to determine which variables should be perturbed at each generation: the random picking method (RP), the majority voting based on the similarity between high quality individuals (MVSH) and the majority voting based on the difference between high quality and low quality individuals (MVDHL). Then, subEDAs + RP, subEDAs + MVSH and subEDAs + MVDHL are tested on several benchmark functions and their algorithmic results are compared with those obtained by EDAs. Our experimental results indicate that subEDAs are able to obtain a comparative result using only a subset of problem variables in the model when compared with traditional EDAs.

1. Introduction

In recent years, an increasing interest has been concentrated on a class of probabilistic and graphical model based evolutionary computational methods, commonly called as estimation of distribution algorithms (EDAs) [1,2]. In EDAs, there is neither crossover nor mutation operator of traditional genetic algorithms and new candidate individuals at the next generation are reproduced through sampling from a probability distribution of promising individuals. Let X be a discrete random variable. A value of X is denoted x. $\mathbf{X} = (X_1, \ldots, X_n)$ will denote a vector of random variables. We will use $\mathbf{x} = (x_1, \ldots, x_n)$ to denote an assignment to the vector of random variables. The joint probability mass function of $\mathbf{x}$ is represented as $p(\mathbf{X} = \mathbf{X})$ or $p(\mathbf{x})$. The pseudocode of the EDAs is described in Algorithm 1. EDAs explicitly identify the dependence among variables by a greedy search method and are able to solve complex combinatory optimization problems with a high convergent reliability and low time consumption. Therefore, EDAs have been widely applied into many real world problems such as image analysis [3], scheduling [4], feature selection [5] and machine learning [6,7].

Algorithm 1 *(Estimation of distribution algorithms).*
Set $t \leftarrow 0$.
do
  If $t = 0$
    Generate an initial population D_0 at random.
    Evaluate population D_0 using an evaluation method.
  Else
    Sampled a $D_{sampled}$ population from the model using a sampling method.
    Evaluate population $D_{sampled}$ using an evaluation method.
  $D_t^{Se} \leftarrow$ Select a set of points from $D_{sampled}$ (when $t = 0$, let D_0) according to a selection method.
  $p_{t+1}(\mathbf{x}) = p(\mathbf{x}|D_t^{Se}) \leftarrow$ Compute a probabilistic model of D_t^{Se} using a learning method.
  $t \leftarrow t + 1$
until Stopping criterion is met.

Several different kinds of EDAs have been proposed so far due to different models for estimating the probability distribution of

* Corresponding author at: Department of Electronic Commerce, South China University of Technology, Guangzhou 510006, China.
E-mail addresses: hlongli@scut.edu.cn, likanye@163.com (H. Li), yihongucla@gmail.com (Y. Hong), cssamk@cityu.edu.hk (S. Kwong).

1568-4946/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.asoc.2010.11.022

promising individuals. Among them is univariate marginal distribution algorithm (UMDA), which assumes all variables to be independent and the probability density of promising individuals is thus estimated as the product of the frequencies of all variables [8]. In mutual information maximization for input clustering (MIMIC), the dependence among variables is considered as a chain and the probability density of promising individuals is calculated using one univariate marginal density and $(n-1)$ pairwise conditional densities [9]. Estimation of Bayesian network algorithm (EBNA) is a well designed EDAs that employs a Bayesian network to capture the dependence among variables [10]. Extended compact genetic algorithm (ECGA), which consists of using a marginal product model to estimate the joint probability distribution of the selected individuals in each generation, was carried out using a greedy forward algorithm [11,12]. Dong and Yao proposed a new EDAs ensemble method named as the NichingEDA framework, which can be regarded as a combination of evolutionary algorithm (EA) and EDAs to overcome the weakness of classical single model based on EDAs [13]. In NichingEDA, the entire population consists of several separated subpopulations. Each subpopulation is a single model based EDAs, which also contains a population of individuals [13]. But how to design or apply other probabilistic models or evolutionary operators to improve the performance and scalability of NichingEDA was not analyzed.

In most existing EDAs, all variables of candidate individuals are perturbed through sampling from a probability distribution of promising individuals. However, it may be unnecessary for EDAs to perturb all variables of candidate individuals at each generation. This is because one variable may be dependent on another variable and all variables may have different saliences even if they are independent. Therefore, only a subset of all variables really function at each generation in EDAs. It is thus of great interest and also promising to explore a novel class of EDAs, termed as subspace estimation of distribution algorithms (subEDAs), in which only a subset of all variables are perturbed at each generation. The goal of this paper is to investigate the advantages of using variables partially in EDAs.

The remainder of this paper is arranged as follows. In Section 2 we consider UMDA and EBNA for discrete optimization problems. The sequence convergence base on EDAs is given in Section 3. In Section 4, we go into details of describing subEDAs. Experimental studies of subEDAs are given in Section 5. Our final conclusions are draw in Section 6 along with future work.

2. Preliminaries

In this paper, we focus on the marginal product models (MPM) and the factorization of the joint probability distribution to EDAs for combinatorial optimization. Two well known examples are the UMDA [8] and EBNA [10]. Here we firstly analyze these algorithms and the learning procedures they employ.

2.1. UMDA

UMDA uses a probabilistic model where all variables are considered independent to estimate the joint probability distribution in each generation. The joint probability distribution of UMDA, which is the simplest MPM of those used by EDAs, is factorized as a product of independent univariate marginal distributions:

$$p_t(\mathbf{x}) = p(\mathbf{x}|D_{t-1}^{Se}) = \prod_{i=1}^{n} p(x_i|D_{t-1}^{Se}). \tag{1}$$

Each univariate marginal distribution is estimated from marginal frequencies:

$$p(x_i|D_{t-1}^{Se}) = p_{t-1,i}(x_i) = \frac{\sum_{j=1}^{N}\delta_j(X_i = x_i|D_{t-1}^{Se})}{N} \tag{2}$$

where N is the selected individuals from D_{t-1} and $p_{t,i}(x_i)$ is the estimated probability of the i th bit of a individual to be x_i in the t generation and

$$\delta_j(X_i = x_i|D_{t-1}^{Se}) = \begin{cases} 1 & \text{if in the } j\text{th case of } D_{t-1}^{Se}, \quad X_i = x_i \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

As one of the earliest version of EDAs, all the variables of UMDA in the probability model are assumed to be independent of one another for all $t \geq 0$. Therefore, the structure of the model for UMDA is fixed and only the univariate marginal probabilities are learned from the data. In addition, the population based incremental learning (PBIL) [14] and the compact genetic algorithm (CGA) [15] belong to this type of models, where the information collected from previous generations is used to update the univariate probabilities. UMDA and other EDAs based on univariate models not only have been applied for binary problems [16–18], but also have been investigated to discrete problems of higher cardinality [19], so their attention is more and more in the field of evolutionary computation.

2.2. EBNA

Similar to Bayesian optimization algorithm (BOA) [20] and factorized distribution algorithm (FDA) [21], EBNA is proposed by Larrañaga et al. [1,10]. EBNA adopts Bayesian network as the probabilistic model which is learned from the database containing the selected individuals at each generation. There are different kinds of EBNA to learn the structure of a Bayesian network [1]. Here, we introduce only EBNA_{BIC} used in our experiments. In general, EBNA_{BIC} searches for the better structure of Bayesian network using search + score method. In this strategy, scoring is achieved by penalized maximum likelihood $\text{BIC}(S, D)$ for a given structure S and a dataset D, called Bayesian information criteria (BIC) [22], denoted by the following equation:

$$\text{BIC}(S, D) = \sum_{i=1}^{n}\sum_{j=1}^{q_i}\sum_{k=1}^{r_i} N_{ijk}\log\frac{N_{ijk}}{N_{ij}} - \frac{1}{2}\log N\sum_{i=1}^{n} q_i(r_i - 1) \tag{4}$$

where the structure S is represented by direct acyclic graphs, n is the number of variables of the Bayesian network, r_i is the number of different values that variable X_i can take, q_i is the number of different values that the parent variables of X_i in the structure S can take, N_{ij} is the number of individuals in D in which the parent variables of variable X_i take their j th value, and N_{ijk} is the number of individuals in D in which variable X_i takes its k th value and the parent variables of the variable i take their j th value [22]. To find the Bayesian network implies solving an optimization problem. This can be done with exhaustive or heuristic search algorithms [23]. Applying EBNA to solve complex problem can bring them nearer to practical use [24]. In the experiments presented in this paper, EBNA uses truncation selection and the number of selected individuals equals half of the population.

3. Sequence convergence base on EDA

Millions of genes exist in a biology body and these genes may have complex relationship as well as different saliences. As a result,

only a subset of all genes function, while other genes are redundant and contribute less for the evolution of the biology at a certain evolutionary stage. This natural phenomenon of the periodical functioning of genes lets us rethink the framework of existing EDAs.

For a vector $\mathbf{a} = (a_1, a_2, \ldots, a_k)$, let $||\mathbf{a}|| = (\sum_{j=1}^{k} a_j^2)^{1/2}$ be the usual Euclidean norm. With this notation, we can define convergence in probability [25,26] in a way to the scalar case:

Definition 1. A sequence of random vectors $\{\mathbf{X}_t = (X_1^t, X_2^t, \ldots, X_k^t); t = 1, 2, \ldots\}$ is said to converge in probability to $\mathbf{X} = (X_1, X_2, \ldots, X_k)$ if for all $\varepsilon, \delta > 0$, $\exists N$, s.t. $\forall\, t \geq N, P\{\,||\mathbf{X}_t - \mathbf{X}\,|| > \delta\} < \varepsilon$. This will be written as either $\mathbf{X}_t \rightarrow {}^{P}\mathbf{X}$ or $plim_{n\rightarrow\infty}\mathbf{X}_t = \mathbf{X}$.

According to the above definition, we can easily know that existing positive integer $n(\varepsilon, \delta, N) > 0$, s.t. when $t > n(\varepsilon, \delta, N)$, we have $P\{|X_i^t - X_i| > \delta(\varepsilon, N), i = 1, 2, \ldots, k\} < \varepsilon$, which means a variable converged.

Definition 2. k sequences of random variables $\{X_1^t\}, \{X_2^t\}, \ldots, \{X_k^t\}$ are called to converge one after another with a certain order, if for all $\varepsilon, \delta > 0, \exists N_{(i)} (i = 1, 2, \ldots, k)$ satisfying $N_{(1)} \leq N_{(2)} \leq \ldots \leq N_{(k)}$, where $(1), (2), \ldots, (n)$ is a random permutation of $1, 2, \ldots, n$, s.t. $\forall t \geq N_{(i)}$, $P\{|X_i^t - X_i| > \delta(\varepsilon, N_{(i)})\} < \varepsilon$.

Lemma 1. *Let X be a random variable, for any given $\varepsilon > 0$, $\exists$ certain $L > 0$, s.t. $P\{|X| > L\} < \frac{\varepsilon}{2}$.*

Proof. Take the sequence $L_1, L_2, \ldots, L_t$, s.t. $lim_{t\rightarrow\infty} L_t = \infty$. we can obtain $P\{\,|X\,| > L_t\} = 1 - P\{\,|X\,| \leq L_t\} \rightarrow 0$. Therefore, for any $\varepsilon > 0$, exists certain positive number N, when $t > N$, we have $P\{|X| > L_t\} < \frac{\varepsilon}{2}$. □

Theorem 1. *Let $g(x_1, \ldots, x_k)$ be a continuous function at every point $(x_1, \ldots, x_k)$ in a domain $\mathbb{R}^k$, if $\mathbf{X}_t \rightarrow {}^{P}\mathbf{X}$, then $g(\mathbf{X}_t) \rightarrow {}^{P}g(\mathbf{X})$.*

Proof. According to Definition 1 and Lemma 1, combined with the property of function continuity, we can easily prove the above conclusions. □

Theorem 2. *Let $g(\mathbf{x})$ be a rational function at every point $\mathbf{x} = (x_1, \ldots, x_k)$ in domain $\mathbb{R}^k$. After using EDA to generate an initial population at random, the variables $\{X_1^t\}, \{X_2^t\}, \ldots, \{X_k^t\}$ of all sampled individuals will finally converge to $X_1, X_2, \ldots, X_k$ one after another with a certain order, s.t. $g(X_1, X_2, \ldots, X_k)$ is the optimal values.*

Proof. Here we only prove the simplest case. Our analysis mainly carried out the Leadingones function [27]. The function is $g(\mathbf{x}) = \sum_{i=1}^{k}\prod_{j=1}^{i} x^{(j)}$, where $x^{(j)} = 0$ or 1. We try to use UMDA as EDA approach to search for the maximum value of Leadingones function. We know that if at the i th generation, $p_{t,i}(1) = 1$, then for $\forall t > t_0, p_{t,i}(1) = 1$ [28]. Combined with the formula (2) and Definition 2, we can easily draw a conclusion that $(\{X_1^t\}, \{X_2^t\}, \ldots, \{X_k^t\})$ of all sampled individuals will finally converge to $(1, 1, \ldots, 1)$ one after another. On the other hand, based on Theorem 1, we obtain the result that $g(1, 1, \ldots, 1)$ is the optimal values of Leadingones function. □

From the above theorem, we observe that variables function periodically and converge one after another with a certain order in EDAs. Usually there are two cases in EDAs where variables function periodically and converge one by one with a certain order. The first case is that variables have a fixed dependence. One example of this case is given in Fig. 1(a), where both the variable X_2 and X_3 are dependent on the variable X_1. Here if the variable X_1 has not been successfully evolved and randomly distributed in the population, all other variables are quite difficult to be valued and effectively evolved. Another case is that the variables are independent and have different saliences. Fig. 1(b) gives an example of this case, where the variables X_1, X_2, X_3 are significantly important and

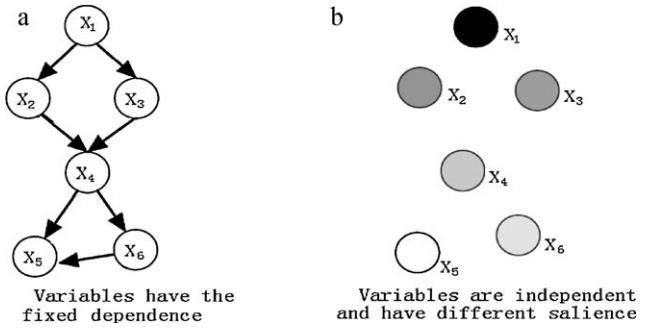

Fig. 1. Two cases where variables function and sequence convergence in EDAs.

contribute more than the variables X_4, X_5, X_6 for the fitness evaluations of candidate individuals. Therefore, in any run of the EDA, some variables will change more than others at different individuals in the evolution. In addition, the order that variables converge depends on the structure learning and sampling methods except for some directed dependency in variables.

To better illustrate the periodical functioning and ordered convergence of variables in EDAs, UMDA is tested on the BinInt problem. The BinInt problem, which has an exponential scaled salience or fitness structure [29,30], is defined as:

$$f(\mathbf{x}) = \sum_{i=1}^{n} 2^i x_i, \qquad x_i \in \{0, 1\}$$

where n is the string length and x_i the alleles. Obviously, variables have significantly different importance and fitness values of all individuals are only dependent on a subset of all variables. Furthermore, the salience of one particular gene is higher than the combined marginal fitness contributions of all the following genes. For example, fitness values of all individuals with the value 1 of the variable X_{20} are larger than the fitness values of all individuals with the value 0 of the variable X_{20}. As a consequence, the variable X_{20} functions at the initial evolutionary stage and converges firstly, then the variable X_{19} follows and the last variable that functions and converges is X_1. This phenomenon shows that the genes converge sequentially from those with the highest salience and finish with those of the lowest salience. The sequential convergence phenomenon was named as domino convergence and the above domino convergence was firstly observed and analyzed [29]. Thierens et al. [30] applied a sequential parameterization approach to model this domino convergence and showed that problems with a non-uniform distribution of the marginal fitness contribution of building blocks typically resulted in a temporal differentiated convergence behavior. In [31], the authors claimed that local domino convergence is a common phenomenon in EDAs. Fig. 2 shows a plot of this convergence behavior. The experiment is done for 20 independent runs. The following two phenomenon can be observed from the evolutionary curves of probabilities of variables shown in Fig. 2(a). First, variables converge one by one in a certain order. Furthermore, the more salient genes are already fully converged while the genes at the other side do not start to converge yet. This phenomenon can be more clearly observed from the generation of each variable of being converged in Fig. 2(b). The domino convergence of variables indicates that the variable X_{20} converges earlier than other variables and the variable X_{19} follows and the variable X_1 converges lastly. The above phenomenon also shows that to perturb only some variables is really beneficial. Second, probabilities of some variables increase and probabilities of other variables decrease at each generation when using UMDA to optimize BinInt problem.

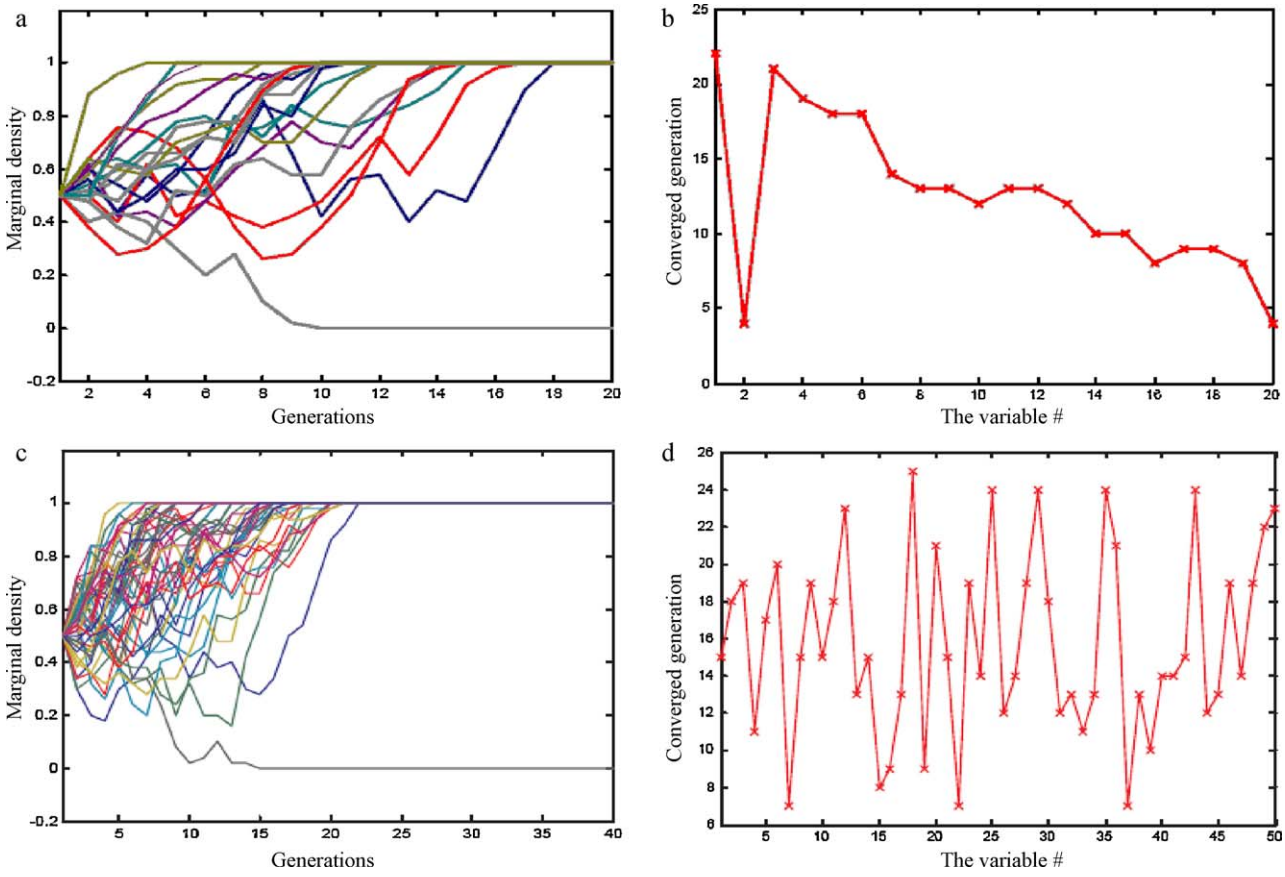

Fig. 2. Ordered functioning and convergence of variables for BinInt problem and OneMax problem.

Apart from BinInt problem, another tested problem is the OneMax problem [32] that is defined as:

$$\sum_{i=1}^{n} x_i, \quad x_i \in \{0, 1\}$$

All variables in the OneMax problem are independent as well as equally salient and the expected proportion of optimal alleles at each generation is equal for the entire string. However, our experimental results shown in Fig. 2(c) and (d) indicate that there still exist the periodically functioning and order convergence of variables. Among all 50 variables, three variables converge at around the 5 th generation and another five variables converge at around the 24 th generation, showing a great difference of convergent order of variables. The above experimental results let us know that only part variables play a role in the run of EDAs. It is desirable to design specialized EDAs to deal with the useful part variables. In fact, the periodical functioning and ordered convergence of variables are quite common phenomenon in EDAs. This maybe is because the time to wait for convergence for the less salient building blocks is limited due to genetic drift in a finite sized population [30,32,33].

The above theorem and two experiments indicated that only a subset of all variables functioned at a certain evolutionary stage in EDAs. In addition, it is well known that more variables usually lead to a slower convergence of EDAs and much more time consumption to learn the dependence among variables. It is therefore of great interest and also promising to explore a novel class of EDAs, in which only a subset of all variables are perturbed at each generation.

4. subEDAs

Problems for which variables converge sequentially are important. It is desirable to design specialized EDAs to deal with this problem explicitly. This section goes into details of describing a novel class of EDAs, termed as subEDAs. In subEDAs, only a subset of all variables are selected and perturbed at each generation and other variables remain unchanged. Before further illustrating about subEDAs, some notations used throughout this paper are given as follows: n is the number of variables; X_i is the i th variable; x_{ai} is the value of the i th variable in the individual x_a; I^H is the selected subpopulation of high quality individuals and I^L is the selected subpopulation of low quality individuals.

4.1. To determine which variables will be perturbed at each generation

As reviewed in Section 3, parts of the variables have been developed for EDAs. To find the determined variables is the key theme in the process of subEDAs. Therefore, one major step of subEDAs is to determine which variables will be perturbed at each generation. In this paper, the following three simple schemes are described into details: the random picking method (RP), the majority voting based on the similarity between high quality individuals (MVSH) and the majority voting based on the difference between high quality and low quality individuals (MVDHL).

In this paper, we first investigate a explicit stochastic scheme, named as RP. Since the initial population is usually obtained at random by sampling a Bernouilli distribution with a parameter value equal to 0.5 in EDAs, the evolved variables can be taken as the natural representation of the current information. The full steps of RP are

given in Algorithm 2, where a float number is randomly generated from [0, 1] and its value is compared with 0.5; if its value is bigger than 0.5, then the variable is selected and perturbed, otherwise the variable is neglected. The key idea is to store important variables which have good solutions. Obviously, RP works to randomly select half of all variables for perturbing at each generation.

Algorithm 2 *(Random picking method (RP)).*

$S = \varnothing$;
for $i = 1 : n$ **do**
  if $rand(1) \geq 0.5$ **then**
    $S = S \bigcup \{X_i\}$;
  end if;
end for;
return S

The second scheme is MVSH. The main idea of MVSH is to detect some similar variables between high quality individuals. The variable X_i is selected and perturbed at this generation if and only if most high quality individuals have the same value of the variable X_i and the variable X_i has not converged. To realize it, two high quality individuals x_a and x_b are randomly selected and their variables are checked. The above steps iterate for turn rounds and the frequency that high quality individuals have the same value of each variable can be approximated. It is worthwhile mentioning that MVSH does not calculate marginal densities of all variables in I^H and only several high quality individuals are randomly chosen and employed for determining which variables will be perturbed at each generation. Algorithm 3 describes the implemented steps of MVSH.

Algorithm 3 *(Majority voting based on the similarity between high quality individuals (MVSH)).*

$S = \varnothing$;
$Count = \vec{0}$;
for $t = 1 : turn$**do**
  $x_a, x_b \leftarrow$ to randomly pick two individuals from I^H;
  for $i = 1 : n$**do**
    if $x_a(i) == x_b(i)$**then**
      $Count(i) = Count(i) + 1$;
    end if
  end for
end for
$Count = \frac{Count}{turn}$;
for $i = 1 : turn$**do**
  if $Count(i) \geq 0.5$ and X_i has not converged **then**
    $S = S \bigcup \{X_i\}$;
  end if;
end for;
return S.

The third scheme is MVDHL. In MVDHL, the variable that can distinguish between high quality individuals and low quality individuals well is selected and perturbed at each generation. Algorithm 4 gives the full steps of MVDHL. It can be observed from Algorithm 4 that MVDHL works with the following steps employed: a high quality individual x_a and a low quality individual x_b are randomly chosen and their variables are compared; the above steps iterate for turn rounds, the probability of each variable's distinguishing capability between high quality individuals and low quality individuals can be calculated; then MVDHL determines which variables will be perturbed using the majority voting technique on the probability of each variable's distinguishing capability between high quality individuals and low quality individuals.

Algorithm 4 *(Majority voting based on the difference between high quality and low quality individuals (MVDHL)).*

$S = \varnothing$;
$Count = \vec{0}$;
for $t = 1 : turn$**do**
  $x_a \leftarrow$ to randomly pick one individual from I^H;
  $x_b \leftarrow$ to randomly pick one individual from I^L;
  for $i = 1 : n$**do**
    if $x_a(i) \neq x_b(i)$**then**
      $Count(i) = Count(i) + 1$;
    end if;
  end for;
end for;
$Count = \frac{Count}{turn}$;
for $i = 1 : turn$**do**
  if $Count(i) \geq 0.5$ and X_i has not converged **then**
    $S = S \bigcup \{X_i\}$;
  end if;
end for;
return S.

It is noted that our proposed three variables selection schemes RP, MVSH and MVDHL may be not the best and only can roughly determine which variables should be perturbed at each generation. The advantage of RP, MVSH and MVDHL is that all of them are very simple to implement with a low time consumption. Existing supervised feature selection algorithms may be a good replacement of RP, MVSH and MVDHL and can identify a better subset of variables for perturbing at each generation. However, all these supervised feature selection algorithms will increase the time consumption of EDAs a lot, thus significantly decrease the interest of our proposed subEDAs. This is because our proposed subEDAs cost much less time consumption to learn the dependence among variables at each generation and finally reduce the overall time consumption of EDAs for solving large scale combinatory optimization problems. In addition, it may be unnecessary to obtain the accurate subset of variables of being perturbed at each generation, because EDAs are a heuristic search method where the chaotic character is also one of the key components for their success. Another thing is that MVSH does not calculate the marginal densities of all variables for checking if the variable has converged at each generation. Instead, marginal densities of all variables are directly copied from those at the above generation.

4.2. Framework of subEDAs

This subsection describes a novel class of EDAs, termed as subEDAs, that combines existing EDAs with our proposed variables selection schemes. Algorithm 5 gives the whole framework of subEDAs. It can be observed from Algorithm 5 that the only difference between subEDAs and EDAs is that the subEDAs estimate the probability density of promising individuals in a selected subspace of all variables and all individuals are perturbed through sampling from the density in this variable subspace, while maintaining other variables unchanged at each generation. The reproduction of new individuals in subEDAs can be described as follows:

$$x_{ik} = \begin{cases} \text{to sample from } P^S(x), & \text{if } X_k \in S, \\ x_{ik}, & \text{otherwise.} \end{cases} \quad (5)$$

where $i = 1 : M$, $k = 1 : n$, M is the population size and $P^S(x)$ is calculated in Algorithm 5. The main advantage of subEDAs is that a lot of time consumption can be saved for learning the dependency among variables and estimating all probability densities from selected high quality individuals. This is because subEDAs only consider a subset of all variables at each generation for learning the probability density of promising individuals. For example, MIMIC calculates n marginal densities and $((n-1)(n-2))/2$ conditional densities and executes $n(n-1)/2$ comparisons for learning the chain structure. However, subEDAs + RP only needs to calculate $n/2$ marginal densities and $(((n/2)-1)((n/2)-2))/2$ conditional densities and execute $((n/2)((n/2)-1))/2$ comparisons for learning the chain structure.

In addition, subEDAs spend much less time on sampling new individuals from the probability density of promising individuals, because subEDAs only perturb a subset of all variables at each generation.

Algorithm 5 *(Subspace estimation of distribution algorithms (subEDAs)).*

(1) $I \leftarrow$ to generate the initial population at random;
(2) $f \leftarrow$ to calculate fitness values of all individuals;
(3) $I^H \leftarrow$ to select some high quality individuals;
(4) $I^L \leftarrow$ to select some low quality individuals;
(5) $S \leftarrow$ to determine which variables will be perturbed;
(6) $P^S(x) \leftarrow$ to compute the probabilistic model of subspace S in I^H;
(7) $I^{new} \leftarrow$ to perturb all individuals in I by sampling from the model $P^S(x)$ in the subspace S;
(8) $I = I^{new}$;
(9) if the stopping criterion is not met, return to (2).

Notes:

- Steps (3) and (4) are unnecessary for subEDAs + RP.
- Step (4) is unnecessary for subEDAs + MVSH.

4.3. The choice of the parameter turn

From Algorithms 3 and 4 we can easily observe that it is important to choose the parameter *turn* to determine which variables should be perturbed at each generation. The effect of the parameter *turn* on the performance of subEDAs is studied in this section. Fig. 3(a), (c) and (b), (d) , respectively, show the relation between *turn* and the number of generations of being converged as well as the optimum values for the OneMax problem. Fig. 3(a), (b) and(c), (d), respectively, choose the population size 300 and 500 to test. The following phenomenon can be observed from Fig. 3(a), (c) that the number of generations of being converged for MVSH tends to certain fixed numeral when the value of the parameter *turn* increased. However, the number of generations of being converged for MVDHL vibrates with the increase in the parameter *turn*. For example, MVSH needs around 18 generations to converge when the parameter *turn* was equal to 10. Its value tends to around 17 if the parameter increased up to 50. For MVDHL the value 2 may be a good choice of the parameter *turn* to obtain the minimal convergence generation. The optimum value of MVSH and MVDHL shown in Fig. 3(b), (d) let us know that the optimal value of MVDHL firstly obtained and finally decreased if the value of the parameter *turn* increased. The optimum curve of MVSH and MVDHL with the population of 300 and 500, respectively, is very similar. In addition, the value 10 may be a good choice of the parameter *turn* of MVSH. Through plenty of experiments for different optimization problems we discover that the parameter *turn* for MVSH set as 10 and the parameter *turn* for MVDHL fixed to 2 are the good choice to analyze the performance of subEDAs. So we fix the above parameter to test the subEDAs in the following experiments.

5. Experimental studies

Several different kinds of EDAs have been proposed so far due to different models for estimating the probability density of promising individuals. To compare subEDAs with standard EDAs, in all the experiments, we use UMDA and EBNA as the representative of EDAs to test because of their simplicity and wide applications. But it should be emphasized that not only UMDA and EBNA, but also any other feasible probabilistic models can be used in subEDAs framework. Therefore, the following eight algorithms were tested in our experiments and their results were compared: UMDA, subUMDA + RP, subUMDA + MVSH, subUMDA + MVDHL, EBNA, subEBNA + RP, subEBNA + MVSH and subEBNA + MVDHL.

We set the same parameter settings in our experiments for all the eight algorithms. As a result of choosing EDAs, all the steps in subEDAs are implemented by the proposed approaches in previous sections. Parameter settings are given as follows. Population sizes of all algorithms were fixed to 500. Among these 500 candidate individuals in the population, the truncation selection scheme is used in the selection task and 250 individuals with the highest fitness values are chosen to construct the subpopulation I^H of high quality individuals and the other 250 individuals are chosen to construct the subpopulation I^L of low quality individuals. The parameter *turn* for MVSH is set as 10 and the parameter *turn* for MVDHL is fixed to 2 in our experiments.

We introduce a set of functions that represent different classes of problems to test the behavior of subEDAs. A more detailed description of each function is given in Appendix A. Here we use three different criteria to compare the algorithms. The convergence reliability, the time consumption and the way in which the selected variables are represented in the subEDAs. In all the experiments and for all the instances of subEDAs, the number of variables n increase from 100 to 1000 with the incremental of 100. All compared algorithms finished if and only if the global optimal solution has been identified or the generation has increased up the maximal value 1000 or the average fitness values of all candidate individuals have been larger than 99% of the best individual in the population. Each experiment is repeated for 20 independent runs and their average result is reported. The mean values and standard errors are computed based on the best solutions that an algorithm has found in multiple runs.

5.1. Convergence reliability of subEDAs

In the analysis of the convergence reliability, we focus on the critical solutions obtained by the EDAs and subEDAs to achieve a predefined convergence results. In the experiments, the goal is to find the optimum in 20 independent runs of consecutive experiments. We investigated the behavior of the algorithms for functions OneMax, WQuad, Trap-5, Plateau, Leadingones, $f_{3\,\mathrm{deceptive}}$.

We begin with a problem size $n = 100$ to obtain the optimal solution of benchmark functions for UMDA, subUMDA + RP, subUMDA + MVSH, subUMDA + MVDHL. For all benchmark functions, UMDA, subUMDA + RP, subUMDA + MVSH, and subUMDA + MVDHL are independently executed for 20 runs and their average best results are shown in Fig. 4. From this figure, we can see that subUMDA is able to obtain a similar or even better solution when compared with the one captured by UMDA. To further elucidate and compare the final solutions obtained by EDAs and subEDAs for all benchmark functions with different numbers of variables, we calculate the final solutions obtained by the compared algorithms. Tables 1–3 only show the final solutions of WQuad, Trap-5, $f_{3\,\mathrm{deceptive}}$ functions obtained by UMDA, subUMDA + RP, subUMDA + MVSH, subUMDA + MVDHL, EBNA, subEBNA + RP, subEBNA + MVSH and subEBNA + MVDHL. It can be observed from Tables 1–3 that for WQuad, Trap-5 and $f_{3\,\mathrm{deceptive}}$ function all compared algorithms identified the global optimal solutions in all 20 independent runs even if the number of variables increased up to 1000. Both Trap-5 and $f_{3\,\mathrm{deceptive}}$ function are deceptive and more difficult to be optimized than OneMax and Plateau [23]. We easily discover from above tables that UMDA, subUMDA + RP, subUMDA + MVSH and subUMDA + MVDHL obtain comparative solutions for WQuad, Trap-5 and $f_{3\,\mathrm{deceptive}}$ function. After comparing the above solutions obtained by UMDA and subUMDA, we can see that subUMDA performs as well as UMDA for WQuad, Trap-5 and $f_{3\,\mathrm{deceptive}}$ function. The same phenomenon are also observed from the final solutions obtained by EBNA, subEBNA + RP, subEBNA + MVSH and subEBNA + MVDHL. On the other hand, subUMDA always achieved better solutions than UMDA and subEBNA always achieved better solutions than EBNA, which can be seen from the boldface

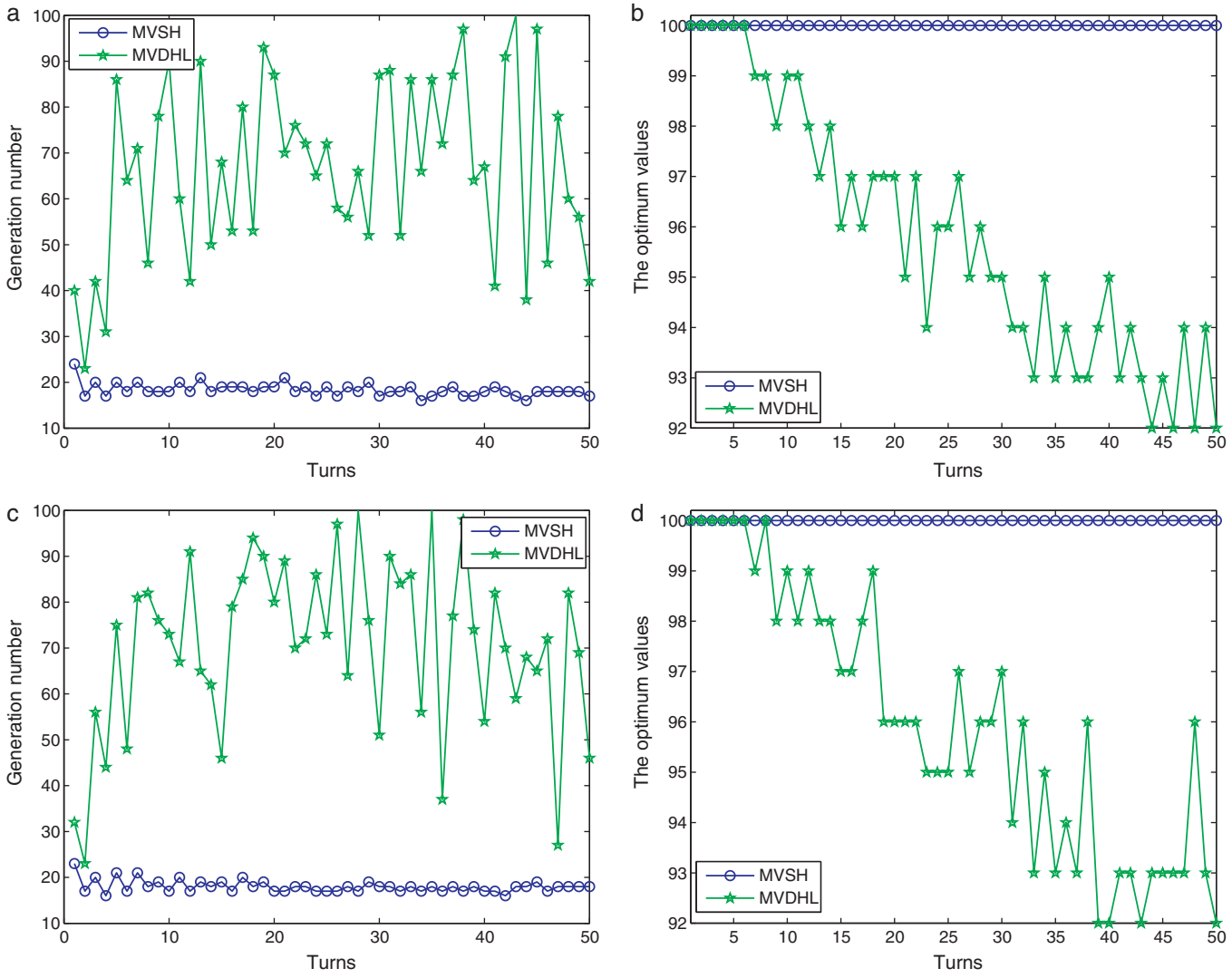

Fig. 3. The effect of the parameter turn to the performance of subEDAs.

mark in Tables 1–3. In addition, the superiority of the solutions obtained by subUMDA or subEBNA over UMDA or EBNA became more significant when the number of variables increased. Thus, we can conclude that subEDAs are often able to obtain a comparative solution when compared with those obtained by standard EDAs.

5.2. Time consumption analysis of subEDAs

As described in the above section, one major advantage of subEDAs is far less time consumption for finishing the search task when compared with EDAs. Therefore, in this subsection the time consumption analysis of subEDAs is studied. Since the overall time

Table 1
Final solutions obtained by the compared algorithms for Weighted Quadratic function.

Algorithms	Average best/std.	$n=100$	$n=200$	$n=300$	$n=400$	$n=500$	$n=600$	$n=700$	$n=800$	$n=900$	$n=1000$
UMDA	Average best	1239	4892	10,906	19,386	30,245	43,364	58,879	76,984	97,080	119,430
	Std.	6.45	32.64	43.87	54.05	76.86	187.56	210.58	251.58	311.50	298.33
subUMDA + RP	Average best	1236	4891	10,930	**19,402**	30,226	43,351	58,842	76,930	96,947	119,250
	Std.	8.97	28.26	48.76	**81.64**	152.46	128.47	206.75	325.38	312.55	284.12
subUMDA + MVSH	Average best	**1241**	**4898**	10,911	19,385	30,353	43,369	58,851	77,173	**97,701**	119,372
	Std.	**10.38**	**24.12**	37.26	90.25	120.84	138.27	200.21	226.64	**286.52**	331.50
subUMDA + MVDHL	Average best	1234	4886	**10,932**	19,389	**30,258**	**43,412**	**59,028**	**77,224**	97,452	**120,342**
	Std.	11.04	31.04	**34.85**	71.38	**169.32**	**129.38**	**203.49**	**174.72**	241.12	**410.36**
EBNA	Average best	1255	4936	10,971	**19,442**	30,256	43,491	59,074	76,988	96,916	119,437
	Std.	6.14	21.38	73.28	**68.20**	98.62	108.73	201.01	231.82	135.02	543.26
subEBNA + RP	Average best	1243	4892	10,918	19,379	30,263	43,221	58,986	76,721	96,784	119,210
	Std.	13.20	28.41	64.58	90.48	81.28	152.36	156.75	262.42	289.26	526.37
subEBNA + MVSH	Average best	**1256**	**4951**	**10,989**	19,409	**30,382**	43,491	58,978	77,115	97,268	119,584
	Std.	**8.46**	**26.76**	**41.82**	69.63	**132.47**	138.42	213.84	382.49	326.20	489.48
subEBNA + MVDHL	Average best	1247	4899	10,940	19,398	30,249	**43,522**	**59,074**	**77,226**	**97,685**	**120,301**
	Std.	9.12	18.84	44.63	104.21	78.46	**162.06**	**221.62**	**294.38**	**250.27**	**482.78**

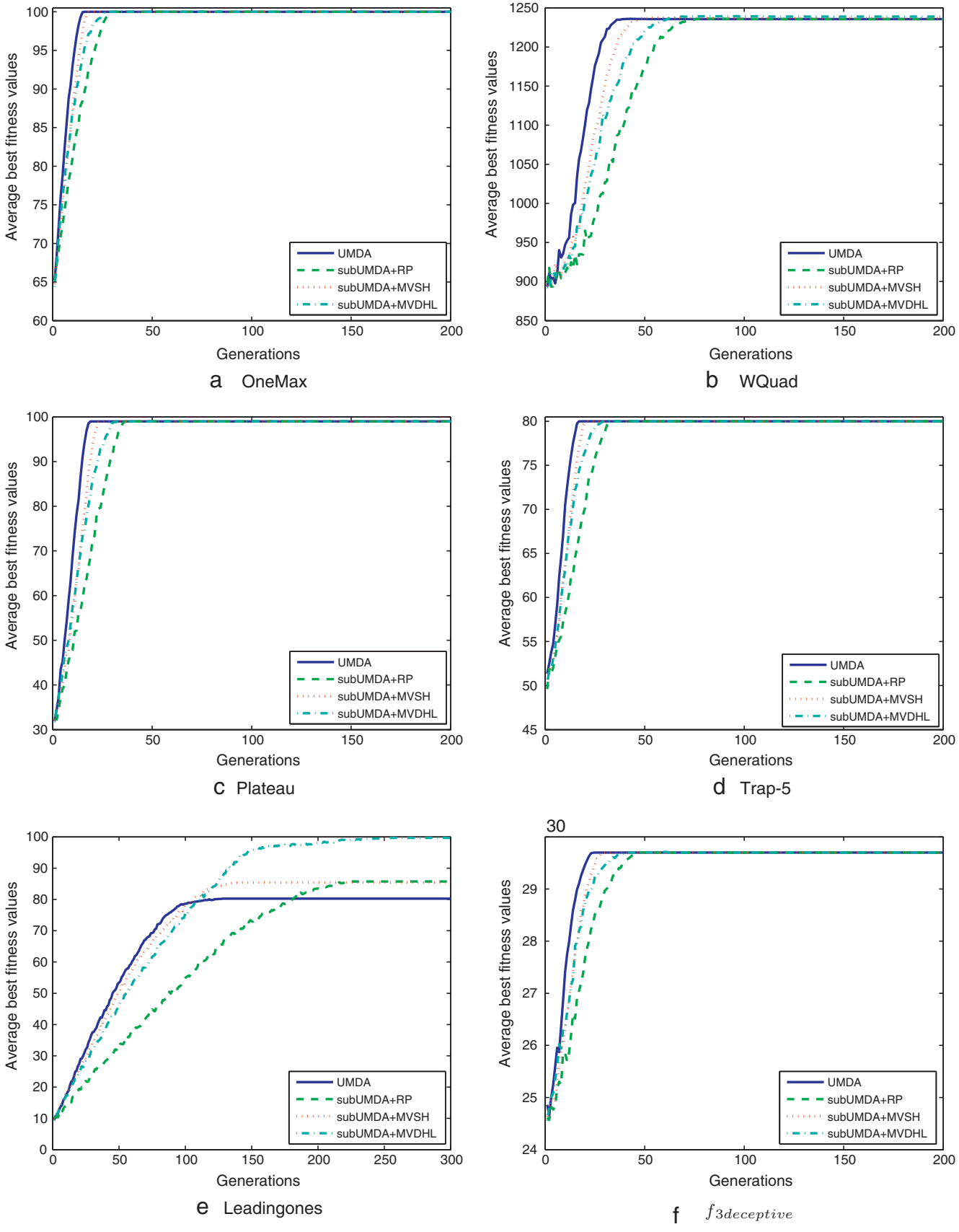

Fig. 4. The convergence reliability of subEDA on test functions with problem size 100.

Table 2
Final solutions obtained by the compared algorithms for Trap-5 function.

Algorithms	Average best/std.	$n=100$	$n=200$	$n=300$	$n=400$	$n=500$	$n=600$	$n=700$	$n=800$	$n=900$	$n=1000$
UMDA	Average best	**79.80**	158.70	237.15	315.02	**391.40**	468.25	540.35	619.55	696.42	769.18
	Std.	**0.42**	1.06	1.42	1.94	**2.07**	2.80	3.58	3.74	5.78	4.58
subUMDA + RP	Average best	79.45	158.45	237. 10	**315.10**	391.30	467.70	**544.75**	**620.20**	9697.15	**773.50**
	Std.	0.64	0.76	1.33	**2.31**	1.39	2.12	**2.24**	**3.37**	4.04	**3.47**
subUMDA + MVSH	Average best	79.50	157.50	235.30	312.80	387.35	464.65	539.70	616.70	690.75	765.65
	Std.	0.67	1.63	1.27	2.10	2.47	2.92	3.92	4.87	4.80	5.62
subUMDA + MVDHL	Average best	79.55	**158.90**	**238.10**	312.97	387.30	**469.12**	543.26	619.23	**698.08**	772.17
	Std.	0.31	**1.82**	**3.28**	5.45	4.63	**5.26**	5.79	6.53	**5.44**	5.81
EBNA	Average best	79.82	159.18	237.38	315.62	392.86	470.16	549.25	**625.86**	701.42	775.25
	Std.	0.32	0.92	0.94	1.54	1.61	2.32	4.82	**2.29**	0.94	3.28
subEBNA + RP	Average best	79.64	160.20	236.22	**316.86**	**394.61**	469.75	**551.72**	624.45	**705.25**	773.88
	Std.	0.68	1.14	2.85	**1.46**	**2.15**	2.11	**4.93**	1.78	**4.83**	5.14
subEBNA + MVSH	Average best	79.48	158.27	237.25	313.23	389.97	468.15	547.48	620.34	698.87	770.60
	Std.	0.84	1.46	1.26	1.93	1.86	2.24	3.78	3.02	5.23	3.36
subEBNA + MVDHL	Average best	**79.94**	**158.90**	**237.95**	315.94	393.45	**470.48**	547.16	618.47	700.20	**775.88**
	Std.	**0.26**	**0.94**	**0.89**	1.48	1.78	**3.16**	3.02	3.16	2.86	**4.89**

consumption of EDAs and subEDAs mainly focus on the sampling and learning time consumption. The time consumption analysis will refer to the sampling and learning time consumption needed by EDAs and subEDAs to find the optimum. All programs are developed by the same configuration and soft environments.

To determine the sampling and learning time consumption to find the optimum needed by EDAs and subEDAs, we start with a population of 50 individuals and the population size is increased by 50 until a maximum population size of 500 is reached. For each execution of the algorithm, we conduct experiments for number of variables $n=300$ and a maximum of 10^5 evaluations are allowed. The results of the experiments are shown in Figs. 5 and 6. Both figures reveal that all algorithms exhibit the similar tendency of time consumption pattern for proposed benchmark functions, which can be seen that the time consumption to obtain the optimal solution becomes more and more with the increase of population size. For example, for the OneMax function, the time consumption of UMDA, subUMDA + MVSH, subUMDA + RP, subUMDA + MVDHL to find the optimal solution is decreased in sequence from high to low. For the WQuad, Plateau, Trap-5, Leadingones and $f_{3\,\mathrm{deceptive}}$ functions, the time consumption of UMDA is most, the following is subUMDA + RP, subUMDA + MVSH and subUMDA + MVDHL. Furthermore, the sampling and learning time consumption of EBNA, subEBNA + RP, subEBNA + MVSH, subEBNA + MVDHL is much more than the UMDA, subUMDA + RP, subUMDA + MVSH, subUMDA + MVDHL, which may be explained that the structure learning of EBNA is more complex. From this analysis we deduce that the overall time consumption of subEDAs is much less than the time consumption of EDAs for solving the benchmark functions.

To illustrate the sampling and learning time consumption of the benchmark functions and to study in more detail the algorithms, we introduce Figs. 7 and 8 which show the sampling and learning time consumption of executions needed by each algorithm in order to find the optimum for solving the benchmark functions with different numbers of variables ranging from 100 to 1000. We can see from Figs. 7 and 8 that the time consumption of subEDAs is much less than the time consumption of EDAs for the same number of variables. Among all these algorithms it is shown that the subEDA + MVDHL is the fastest to reach the optimum for the benchmark functions. Moreover, the superiority of subEDAs over EDAs becomes more significant when the number of variables increase.

5.3. The number of selected variables

To better demonstrate the effectiveness of subEDAs, we fix the number of variables n to 1000 for all tested benchmark problems and study the number of selected variables for perturbing at each generation. The experimental results are given in Fig. 9. It can be observed from Fig. 9 that only a subset of all variables are perturbed in subEDAs at each generation. For example, around 522 variables are selected for perturbing in subUMDA + RP and 509 variables for perturbing in subEBNA + RP, 481 variables for perturbing in subUMDA + MVSH and 600 variables for perturbing in subEBNA + MVSH, 336 variables for perturbing in subUMDA + MVDHL and 328 variables for perturbing in subEBNA + MVDHL at the 60 th generation when solving OneMax optimization problems using subEDAs. The second phenomenon that can be observed from Fig. 9 is the number of selected individuals in subEDAs + MVSH firstly increased from around 608 to

Table 3
Final solutions obtained by the compared algorithms for Coldberg's deceptive function.

Algorithms	Average best/std.	$n=100$	$n=200$	$n=300$	$n=400$	$n=500$	$n=600$	$n=700$	$n=800$	$n=900$	$n=1000$
UMDA	Average best	29.70	59.45	90.05	119.75	**149.55**	180.35	210.15	239.75	270.45	300.95
	Std.	0.15	0.44	0.96	1.10	**2.11**	2.82	3.10	3.41	4.62	3.45
subUMDA + RP	Average best	29.62	59.21	90.12	**121.46**	149.21	181.46	**213.42**	239.90	271.68	301.34
	Std.	0.08	0.65	1.23	**0.98**	1.85	2.44	**2.56**	3.13	3.88	2.86
subUMDA + MVSH	Average best	29.70	**59.82**	**91.83**	120.23	148.74	180.88	212.28	**243.74**	274.20	303.25
	Std.	0.17	**0.63**	**1.14**	1.34	1.23	1.73	2.26	**2.87**	3.24	4.46
subUMDA + MVDHL	Average best	**29.71**	59.56	91.05	120.64	148.96	**181.64**	212.66	241.50	**276.43**	**308.22**
	Std.	**0.22**	0.38	1.83	1.25	1.56	**2.21**	3.28	2.75	**4.21**	**3.27**
EBNA	Average best	29.70	60.67	94.62	124.82	149.62	**184.26**	213.42	245.06	276.62	**310.10**
	Std.	0.12	0.38	1.10	0.90	1.86	**2.32**	2.47	2.24	3.21	2.86
subEBNA + RP	Average best	29.88	60.05	93.50	123.25	**151.46**	182.10	**215.28**	**246.34**	**277.15**	306.82
	Std.	0.06	0.42	0.95	1.02	**2.10**	1.84	**2.10**	**2.47**	**3.48**	3.18
subEBNA + MVSH	Average best	30.10	61.27	93.32	**125.62**	150.20	181.82	212.84	243.20	274.20	304.46
	Std.	0.11	0.51	1.23	**1.34**	1.62	1.67	1.98	3.10	2.96	4.01
subEBNA + MVDHL	Average best	**30.62**	60.82	**94.24**	124.40	150.43	180.64	213.26	242.64	2775.81	308.25
	Std.	**0.16**	0.34	0.87	2.01	1.65	2.10	3.02	2.84	3.75	3.12

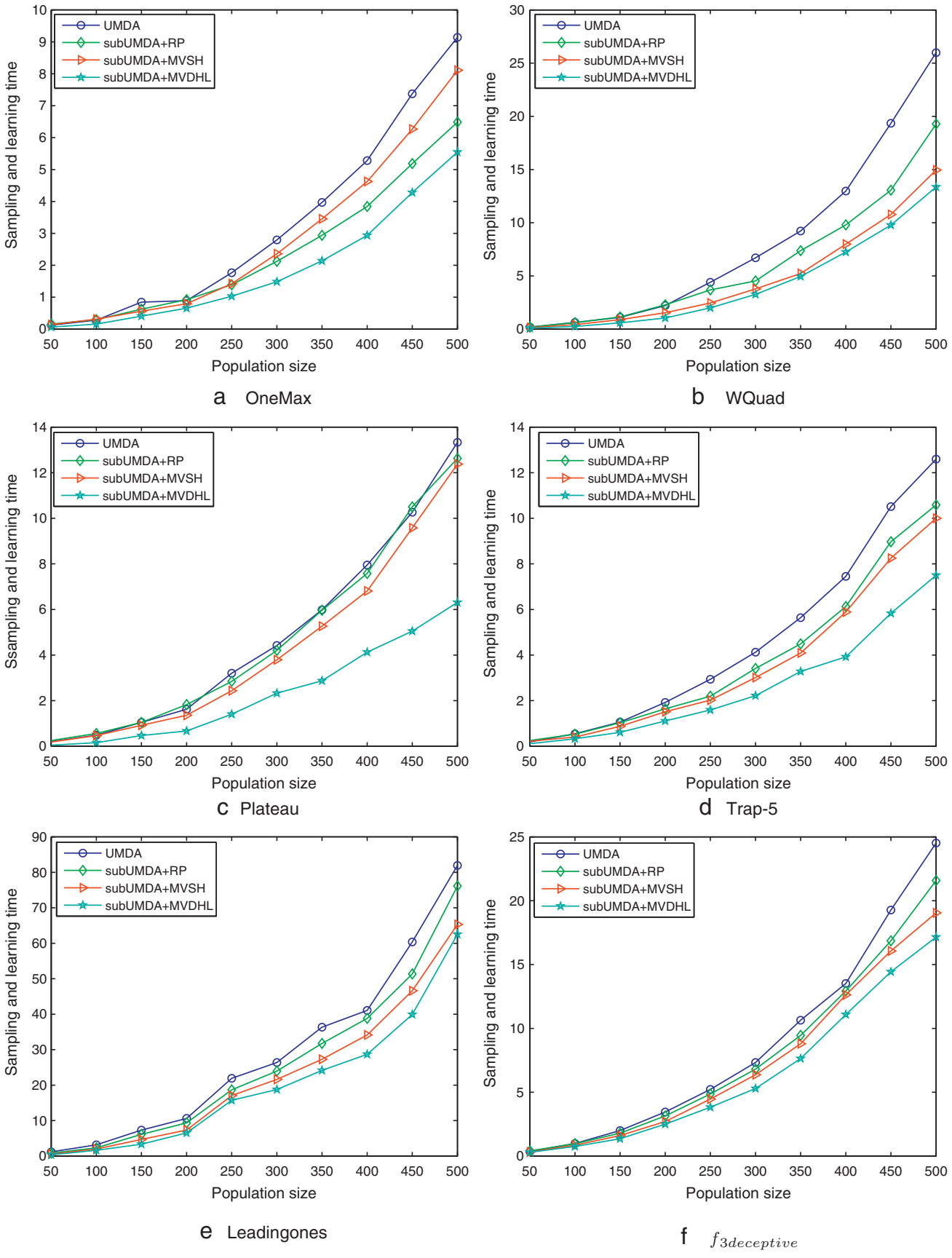

Fig. 5. The time consumption analysis about UMDA and subUMDA for benchmark functions.

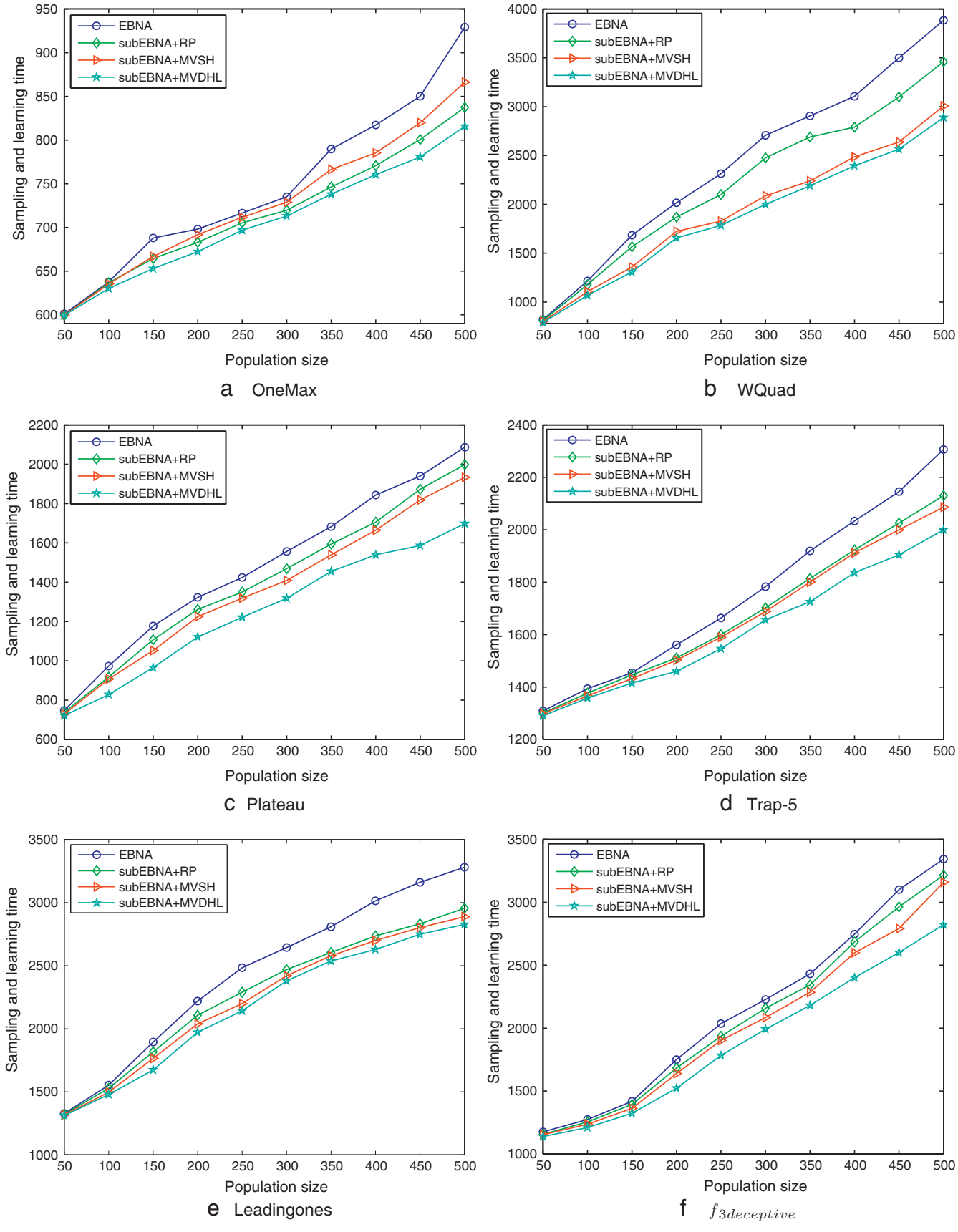

Fig. 6. The time consumption analysis about EBNA and subEBNA for benchmark functions.

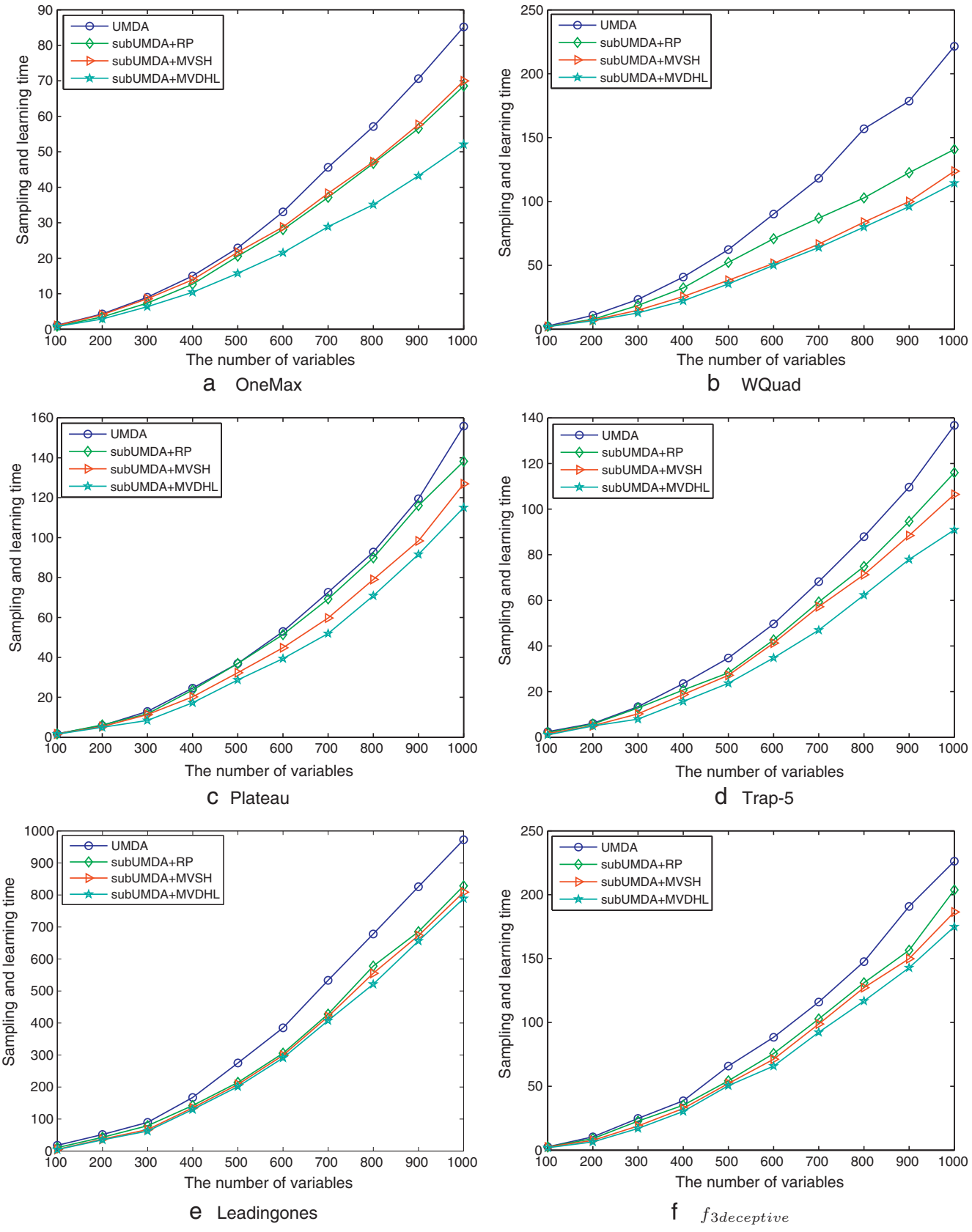

Fig. 7. The time consumption of UMDA and subUMDA based on variables for benchmark functions.

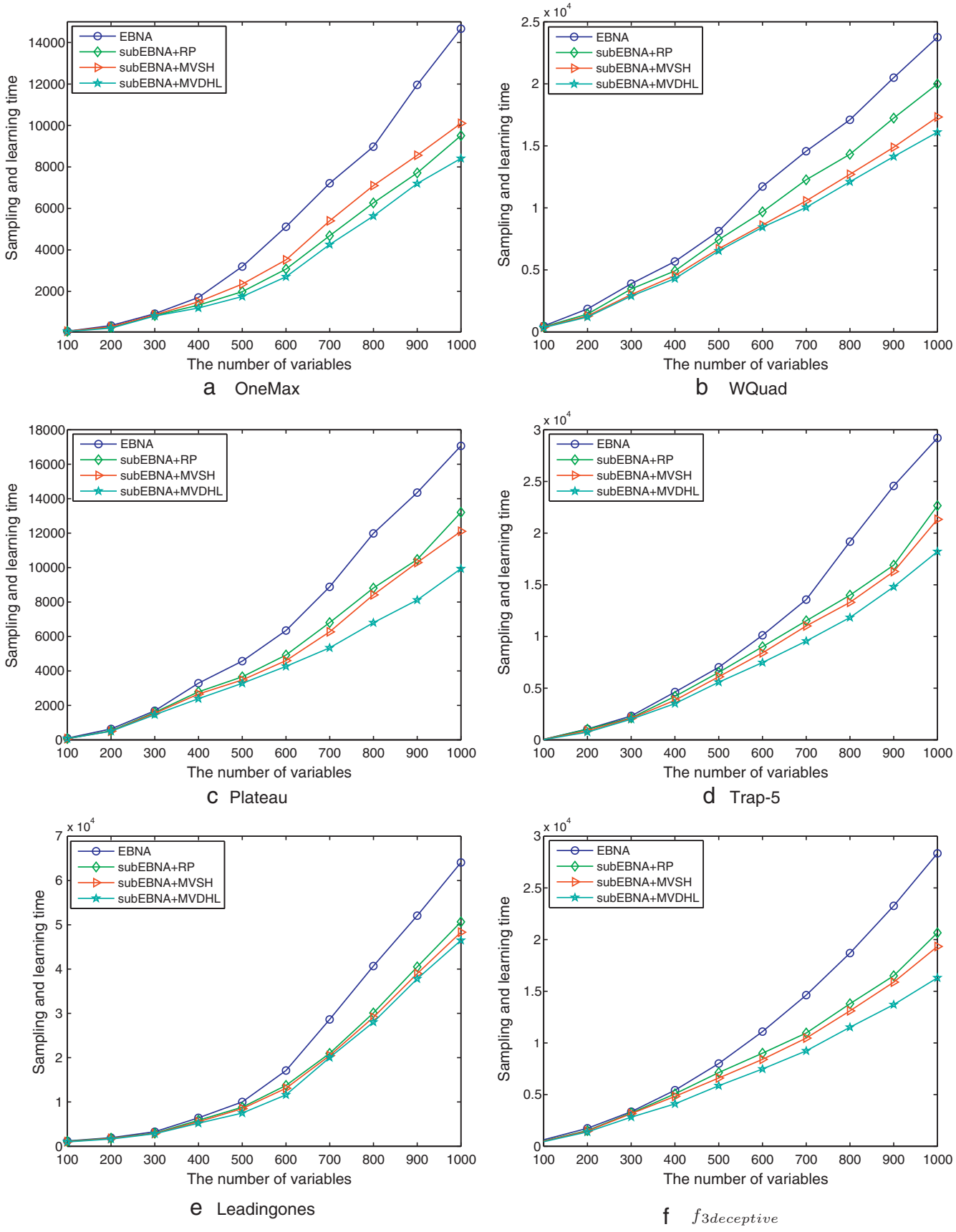

Fig. 8. The time consumption of EBNA and subEBNA based on variables for benchmark functions.

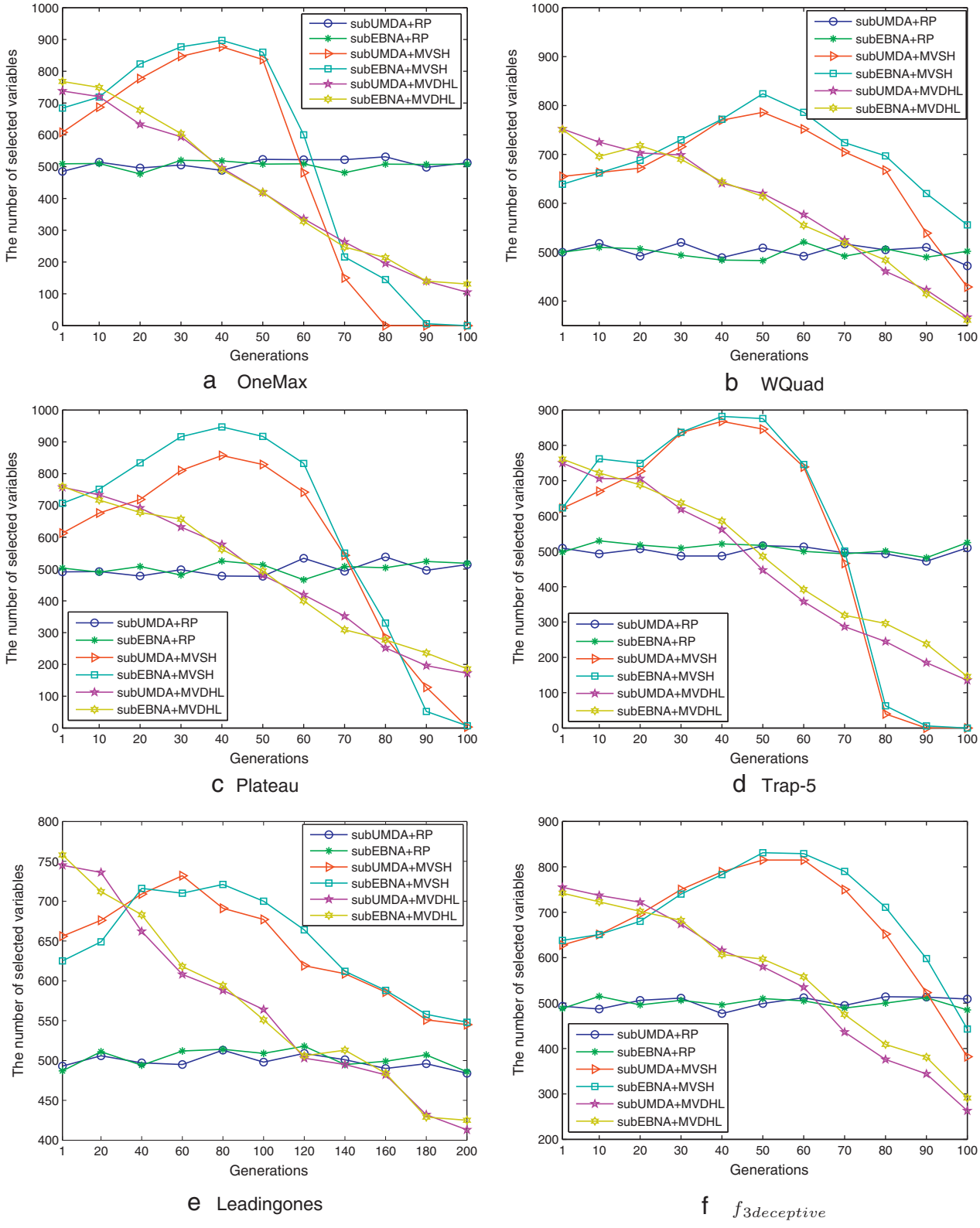

Fig. 9. The number of selected variables at each generation.

877 and finally decreased to 0 at the 80 th generation. Unlike the subEDAs + MVSH, the number of selected variables for perturbing at each generation decreased throughout the whole evolutionary process in subEDAs + MVDHL. Another phenomenon that can be observed from Fig. 9 is the number of selected variables for perturbing in subEDAs + MVDHL was less than the number of selected variables for perturbing in subEDAs + MVSH at the beginning of the evolution and the number of selected variables for perturbing in subEDAs + MVDHL is larger than the number of selected variables for perturbing in subEDAs + MVSH after a certain number of generations. The above three phenomena reveal that the learning time of subEDA + MVDHL may be faster than

subEDA + MVSH or subEDA + RP because of the perturbing variables of subEDA + MVDHL much less with the increase in the number of generations. Besides what you see, the phenomena let us know that only a subset of all variables are selected and perturbed at each generation in subEDAs.

6. Conclusion and future work

In this paper, we have studied the phenomenon of periodical functioning and ordered convergence of variables in EDAs. Inspired from these phenomenon, we have proposed a novel class of EDAs named as subEDAs, which is designed to enhance the sampling and learning efficiency of EDAs. In subEDAs, only part of all variables are selected and perturbed at each generation, while other variables remain unchanged. We have gone into details of describing three variables selection schemes in subEDAs to determine which variables should be perturbed at each generation: RP, MVSH and MVDHL. The superiority of subEDAs over EDAs and the feasibilities of RP, MVSH and MVDHL for determining which variables will be perturbed at each generation have been confirmed by the systematic experimental results for several benchmark functions.

Further research should be done in order to improve the performance of subEDAs, particularly on the theoretical analysis of structure learning about subEDAs. The research level will be improved to give an analytic or probabilistic analysis of proposed method in the subsequent study. Furture work on subEDAs also includes applying other probabilistic models to optimize the scalability of part variables. The desired results are expected in near future.

Acknowledgements

The authors thank the anonymous reviewers whose constructive suggestions and corrections have improved the quality of the paper. This work is supported by Hong Kong RGC GRF Grant 9041353 (CityU 115408), the Fundamental Research Funds for the Central Universities, SCUT (2009ZM0081), NSFC (10826053, 60825306, and U0735004) and GDSF (07118074).

Appendix A.

Benchmark functions:

(1). OneMax function: $f(\mathbf{x}) = \sum_{i=1}^{n} x_i,\ x_i \in \{0, 1\}$.

(2). Weighted Quadratic (WQuad) function: $f(\mathbf{x}) = \sum_{i=1}^{n/2} i(0.9 - 0.9(x_{2i-1} + x_{2i}) + 1.9x_{2i}x_{2i-1}),\ x_{2i-1}, x_{2i} \in \{0, 1\}$.

(3). Plateau function: $f(\mathbf{x}) = \sum_{i=1}^{n/3} g(x_{3i-2}, x_{3i-1}, x_{3i})$, where $g(\cdot) = \begin{cases} 3 & \text{if } x_{3i-2} = 1, x_{3i-1} = 1, x_{3i} = 1; \\ 0 & \text{otherwise;} \end{cases}$

(4). Trap-5 function: $f(\mathbf{x}) = \sum_{i=1}^{n/5} g(x_{5i-4}, x_{5i-3}, x_{5i-2}, x_{5i-1}, x_{5i})$, $u = x_{5i-4} + x_{5i-3} + x_{5i-2} + x_{5i-1} + x_{5i}$, $g(\cdot) = \begin{cases} 5 & \text{if } u = 5; \\ 4 - u & \text{otherwise;} \end{cases}$

(5). Leadingones function: $f(\mathbf{x}) = \sum_{i=1}^{n} \prod_{j=1}^{i} x_j,\ x_j \in \{0, 1\}$.

(6). Goldberg's deceptive ($f_{3\,\text{deceptive}}$) function: $f_{3\,\text{deceptive}}(\mathbf{x}) = \sum_{i=1}^{i=n/3} f_{dec}^{3}(x_{3i-2}, x_{3i-1}, x_{3i})$, where f_{dec}^{3} is defined as: $f_{dec}^{3}(u) = \begin{cases} 0.9, & \text{for } u = 0 \\ 0.8, & \text{for } u = 1 \\ 0.0, & \text{for } u = 2 \\ 1.0, & \text{for } u = 3 \end{cases}$, here $u = (x_{3i-2} + x_{3i-1} + x_{3i})$.

References

[1] P. Larrañaga, J.A. Lozanon, Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation, Kluwer, Norwell, MA, 2002.
[2] P. Pelikan, Hierachical Bayesian Optimization Algorithm: Toward a New Generation of Evolutionary Algorithms, Springer, 2005.
[3] E. Bengoetxea, P. Larranaga, I. Bloch, A. Perchant, C. Boeres, Inexact graph matching by means of estimation of distribution algorithms, Pattern Recognition 35 (12) (2002) 2867–2880.
[4] A. Petrovski, S. Shakya, J. McCall, Optimising cancer chemotherapy using an estimation of distribution algorithm and genetic algorithms, in: Genetic and Evolutionary Computation Conference (GECCO 2006), 2006, pp. 413–418.
[5] Y. Hong, S. Kwong, Y. Chang, Q. Ren, Unsupervised feature selection using clustering ensembles and population based incremental learning algorithm, Pattern Recognition 41 (9) (2008) 2742–2756.
[6] R. Michalski, Learnable evolution model: evolutionary processes guided by machine learning, Machine Learning 38 (2000) 9–40.
[7] Y. Hong, S. Kwong, H. Xiong, Q. Ren, Data clustering using virtual population based incremental learning algorithm with similarity matrix encoding strategy, in: Genetic and Evolutionary Computation Conference (GECCO 2008), 2008, pp. 471–472.
[8] H. Mühlenbein, G. Paaß, From recombination of genes to the estimation of distributions: 1. binary parameters, in: Parallel Problem Solving Inspired from Nature (PPSN 1996), 1996, pp. 178–187.
[9] J.S. De Bonet, C.L. Isbell, P. Viola, Mimic: finding optima by estimating probability densities, in: Advances in Neural Information Processing Systems, 1997, p. 424.
[10] P. Larrañaga, R. Etxeberria, J.A. Lozano, J.M. Peña, Combinatiorial optimization by learning and simulation of Bayesian networks, in: Proceedings of the sisteenth conference on uncertainty in artificial intelligence, Stanford, 2000, pp. 343–352.
[11] G.R. Harik, F.G. Lobo, K. Sastry, Linkage learning in via probabilistic modeling in the Extended Compact Genetic Algorithm (ECGA), Technical Report 99010, IlliGAL Technical Report, 1999.
[12] R. Santana, P. Larrañaga, J. Lozano, Learning factorizations in estimation of distribution algorithms using affinity propagation, Technical Report EHU-KZAA-IK-1/08, Department of Computer Science and Artificial Intelligence, University of the Basque Country, January 2008.
[13] W.S. Dong, X. Yao, NichingEDA: utilizing the diversity inside a population of EDAs for continuous optimization, in: 2008 IEEE Congress on Evolutionary Computation (CEC 2008), 2008, pp. 1260–1267.
[14] S. Baluja, Population-based incremental learning: a method for integrating genetic search based function optimization and competitive learning, Technical Report CMU-CS-94–163, Carnegie Mellon University, Pittsburgh, PA, 1994.
[15] G.R. Harik, F.G. Lobo, D.E. Goldberg, The compact genetic algorithm, IEEE Transactions on Evolutionary Computation 3 (4) (1999) 287–297.
[16] C. González, J.A. Lozano, P. Larrañaga, Analyzing the PBIL algorithm by means of discrete dynamical systems, Complex Systems 12 (4) (2001) 465–479.
[17] Q. Zhang, On stability of fixed points of limit models of univariate marginal distribution algorithm and factorized distribution algorithm, IEEE Transactions on Evolutionary Computation 8 (1) (2004) 80–93.
[18] S. Shakya, J. McCall, D. Brown, Using a Markov network model in a univariate EDA: an empirical cost-benefit analysis, in: H.-G. Beyer, U.-M. OReilly (Eds.), Proceedings of Genetic and Evolutionary Computation Conference GECCO-2005, ACM Press, Washington, DC, USA, 2005, pp. 727–734.
[19] R. Santana, P. Larrañaga, J.A. Lozano, Side chain placement using estimation distribution algorithms, Artificial Intelligence in Medicine 39 (1) (2007) 49–63.
[20] M. Peiikan, et al., BOA: The Bayesian optimization algorithm, Proceedings of the Genetic and Evolutionary Computation Conference 1 (1999) 625–632.
[21] H. Mühliihlenbein, T. Mahnig, FDA – a scalable evolutionary algorithms for the optimization of additively decomposed functions, Evolutionary Computation 7 (4) (1999) 353–376.
[22] H. Handa, O. Katai, Estimation of Bayesian network algorithm with GA searching for better network structure, in: IEEE Int. Conf. Neural Networks & Signal Processing, 2003, pp. 436–439.
[23] C. Echegoyen, R. Santana, J.A. Lozano, P. Larrañaga, The impact of probabilistic learning algorithms in EDAs based on Bayesian networks, in: Linkage in Evolutionary Algorithms, Studies in Computational Intelligence Series, 2008, pp. 109–139.
[24] A. Mendiburu, J. Lozano, J.M. Alonso, Parallel implementation of EDAs based on probabilistic graphical models, IEEE Transactions on Evolutionary Computation 9 (4) (2005) 406–423.
[25] D. Aldous, Discrete Probability and Algorithms, Springer-Verlag, New York, 1995.
[26] R.M. Gray, L.D. Davisson, Random Processes: A Mathematical Approach for Engineers, Prentice-Hall, Englewood Cliffs, NJ, 1986.
[27] G. Rudolph, Finite Markov chain results in evolutionary computation: a tour d'horizon, Fundamenta Informaticae 35 (1998) 67–89.
[28] T.S. Chen, K. Tang, G.L. Chen, X. Yao, On the analysis of average time complexity of estimation of distribution algorithms, in: IEEE Congress on Evolutionary Computation, CEC 2007, 2007, pp. 453–460.
[29] M. Rudnick, Genetic algorithms and fitness variance with an application to automated design of artificial nueral networks, PhD Thesis, Oregon Graduate Institute of Science and Technology, 1992.

[30] D. Thierens, D.E. Goldberg, A.G. Pereira, Domino convergence, drift, and the temporal-salience structure of problems, in: The 1998 IEEE International Conference on Computational Intelligence, 1998, pp. 535–540.
[31] Y. Hong, Q.S. Ren, J. Zeng, Unit bit importance evolutionary algorithm, Soft Computing: A Fusion of Foundations, Methodologies and Applications 11 (2) (2006) 115–122.
[32] J.L. Shapiro, Drift and scaling in estimation of distribution algorithms, Evolutionary Computation 13 (1) (2005) 99–123.
[33] Y. Hong, Q. Ren, J. Zeng, Genetic drift in univariate marginal distribution algorithm, in: Genetic and Evolutionary Computation Conference (GECCO 2005), 2005, pp. 745–746.