An Estimation of Distribution Algorithm With Cheap and Expensive Local Search Methods

Aimin Zhou, *Member, IEEE*, Jianyong Sun, and Qingfu Zhang, *Senior Member, IEEE*

***Abstract*—In an estimation of distribution algorithm (EDA), global population distribution is modeled by a probabilistic model, from which new trial solutions are sampled, whereas individual location information is not directly and fully exploited. In this paper, we suggest to combine an EDA with cheap and expensive local search (LS) methods for making use of both global statistical information and individual location information. In our approach, part of a new solution is sampled from a modified univariate histogram probabilistic model and the rest is generated by refining a parent solution through a cheap LS method that does not need any function evaluation. When the popula-tion has converged, an expensive LS method is applied to improve a promising solution found so far.**

***Index Terms*—Distribution information, estimation of distribution algorithm (EDA), global optimization, location information, univariate marginal distribution algorithm (UMDA).**

I. Introduction

THIS paper considers the continuous box-constrained global optimization problem

$$\begin{aligned} \min \quad & f(x) \\ \text{s.t.} \quad & x \in [a_i, b_i]^n \end{aligned} \tag{1}$$

where $x = (x_1, x_2, \ldots, x_n)^T \in R^n$ is a decision vector, $[a_i, b_i]^n$ is the feasible region of the search space and $a_i \leq x_i \leq b_i$ for $i = 1, 2, \ldots, n$, and $f : R^n \to R$ is the objective function.

Modern heuristics such as evolutionary algorithms (EAs) [1]–[3] have been attracting much attention for tackling global optimization problems. Estimation of distribution algorithms (EDAs) [4], [5] is an EA paradigm. Unlike traditional EAs, EDAs do not use crossover or mutation operators to generate new trial solutions. Instead, they extract global statistical information from solutions visited so far and build a probabilistic model for modeling distribution of promising solutions. New trial solutions are sampled from the model built. Variable linkage information is considered very important for optimization. In EDAs, variable linkages are often represented by a Bayesian network [6], which can be obtained through various machine learning and statistical-learning techniques [7], [8]. EDAs have been applied to many discrete and continuous optimization problems and more details of EDAs are referred to [6] and [9]–[12].

With proper Bayesian network models, EDAs are able to detect variable linkages and sample high-quality solutions. Based on variable linkages being used, Bayesian network models in EDAs can be categorized into the following three classes [9], [10].

1) *Univariate Models:* It assumes that decision variables are independent. The joint probability of the variables is fully factorized as

$$P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i).$$

Examples of such algorithms include PBIL [13], UMDA [4], IDEA [14], cGA [15], and UMDA$_c^G$ [16].

2) *Bivariate Models:* Pairwise variable interactions are considered as

$$P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i | x_{\pi_i})$$

where x_{π_i} is the single parent variable of x_i. Examples include MIMIC [17], MIMIC$_c^G$ [16] and the algorithm based on optimal dependency-tree [18].

3) *Multivariate Models:* Multivariate interactions are considered as

$$P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i | \Pi_i)$$

where Π_i is a set of parent variables of x_i. Examples include FDA [19], BOA [20], EBNA [21], multivariate IDEA [14], CMA-ES [22], EGNA [10], EMNA [10], and others.

Digital Object Identifier 10.1109/TEVC.2014.2387433

Simple models are easy to implement but insufficient for complicated problems [9]. To build a good multivariate model, a large population is required and the learning process (including network structure learning and parameter learning) can be very time consuming [23], [24]. In fact, the learning of Bayesian network itself is $\mathcal{NP}$-hard in some cases. Therefore, it could be very difficult to guarantee the modeling quality. Moreover, multivariate models may overfit and thus mislead the search [25]. To deal with this issue, some efforts have been made to use mixture models [14], [26], [27] and latent variable models [28]–[31].

In conventional EAs, crossover and mutation operators are employed for generating new trial solutions. Major components of new solutions are directly copied from their parents. So, location information of parent solutions is directly used.

In contrast, EDAs work with probabilistic models, while global distribution information of a population is used to generate new solutions.

As argued in [32], solutions generated by crossover and mutation operators may be close to the parents but far away from other promising solutions, whereas EDAs cannot directly control similarities between new solutions and parent solutions. Thus, an efficient EA should make use of both global distribution information and location information of promising solutions. Some possible ways are as follows.

1) Hybridization of EDA and EA for new solution reproduction. In [33], an EDA was combined with a classical genetic algorithm for multiobjective optimization, in which some solutions are sampled from probability models and the others are generated by crossover and mutation operators. **How to efficiently combine different algorithms is a major issue in hybridization strategies [34].**
2) Use of local search (LS) in EDAs [23], [35]. Solutions sampled from models are refined by LS methods.
3) Guided mutation [32]. With guided mutation, part of a new solution is generated by a crossover/mutation operator and the other part is generated by an EDA operator [32], [36], [37]. The DE/EDA method proposed in [38] uses a very similar idea.

In this paper, we propose to use both global statistical information and location information of visited solutions by combining EDA with both cheap and expensive LS. We call this new approach as a hybrid EDA with cheap and expensive LS (EDA/LS). The main characteristics of EDA/LS can be summarized as follows.

1) A modified univariate histogram marginal model, variable-width histogram (VWH), is proposed and applied. **Unlike other EDAs which build models over the whole search space, the VWH model mainly focuses on promising areas.**
2) A local surrogate model based on quadratic approximation is utilized as a cheap LS which does not need any function evaluation (FE). The local surrogate model makes use of individual location information by refining some components of the individuals. A parameter is introduced to balance the contributions of the global information from the VWH model and the local information from the local surrogate model.
3) A modified Powell method [39], [40] is used as an expensive LS. It improves some promising solutions when the population has been considered convergence according to an *ad hoc* convergence detection strategy.

II. Algorithm

A. Algorithm Description

At each generation t, EDA/LS maintains:

1) a population of N solutions, pop $= \{x^1, x^2, \ldots, x^N\}$;
2) their objective function values: $f^1, f^2, \ldots, f^N$.

EDA/LS is presented in Algorithm 1. Its components are explained as follows.

1) *Population Initialization:* In line 1, N solutions are uniformly sampled from $\Pi_{i=1}^{n}[a_i, b_i]$ to form the initial population.
2) *Stopping Condition:* EDA/LS stops when the number of FEs *fe* reaches the preset maximum FE in line 4.
3) *Population Selection:* In line 17, the best N solutions selected from the new solutions and the parent solutions form the population of the next generation.
4) *Population Sorting:* Before cheap and expensive LS, individuals are sorted in an ascending order in terms of their objective function values in lines 5 and 21.
5) *Offspring Reproduction:* A candidate solution is firstly sampled from the probabilistic model in line 8. **Then, each component is replaced by that of the optimal solution derived from the local surrogate model with probability P_c in lines 11–13.** Finally, it is repaired if necessary in line 14.

The details of the other procedures, modeling and sampling, cheap LS, and expensive LS, are discussed in the following.

B. Probabilistic Model Building and Sampling

In this section, univariate histogram marginal models are introduced firstly and then a new histogram model is proposed.

1) Univariate Histogram Marginal Models: Univariate marginal distribution algorithms (UMDAs) with histogram marginal models might be the simplest EDAs for continuous optimization. The joint probability distribution of a general UMDA can be formulated as a product of univariate marginal distributions as

$$P(x) = \prod_{i=1}^{n} P_i(x_i)$$

Algorithm 1: EDA/LS Framework

```
Input:
  1) $N$: population size.
  2) $FE$: maximum number of FE.
  3) $M$: number of bins used in the VWH model.
  4) $P_b$: percentage of best solutions used for local search.
  5) $P_c$: probability to use the location information.
  6) $\theta$: Convergence threshold.
Output: best solution and its objective value.
$pop \leftarrow initialize()$.                                  /* initialize population */
$fe \leftarrow N$.                                              /* set evaluation count */
$t_e \leftarrow 0$.      /* set generation count after the last expensive local search */
while $fe < FE$ do
    $pop \leftarrow sort(pop, \text{'ascend'})$.                       /* sort population */
    $P(X) \leftarrow model(pop, M)$.                                     /* build model */
    for $i \leftarrow 1$ to $N$ do
        $y^i \leftarrow sample(P(X))$.                                  /* sample model */
        $k \leftarrow random\_select\{2, 3, \ldots, \lfloor P_b N \rfloor - 1\}$.          /* select a parent */
        for $j \leftarrow 1$ to $n$ do
            if $rand() < P_c$ then
                $y_j^i \leftarrow cheap\_ls((x_j^{k-1}, f^{k-1}), (x_j^k, f^k), (x_j^{k+1}, f^{k+1}))$.   /* cheap local search */
            end
            $y_j^i \leftarrow \begin{cases} 0.5(x_j^i + a_j) \text{ if } y_j^i < a_j \\ y_j^i \quad\quad\quad\;\; \text{if } a_j \le y_j^i \le b_j \\ 0.5(x_j^i + b_j) \text{ if } y_j^i > b_j \end{cases}$.   /* check feasibility */
        end
    end
    $pop \leftarrow select(\{x^1, x^2, \ldots, x^N\} \cup \{y^1, y^2, \ldots, y^N\})$.   /* select next generation */
    $fe \leftarrow fe + N$.                                     /* set evaluation count */
    $t \leftarrow \frac{fe}{N}$.                            /* set current generation count */
    if $converge(\theta, t, t_e)$ then
        $pop \leftarrow sort(pop, \text{'ascend'})$.                   /* sort population */
        $k \leftarrow random\_select\{1, 2, \ldots, \lfloor P_b N \rfloor\}$.   /* select a solution for expensive local search */
        $\{x^*, fl\} \leftarrow expensive\_ls(x^k, \lfloor 0.5(FE - fe) \rfloor)$.   /* expensive local search */
        $fe \leftarrow fe + fl$.                                /* set evaluation count */
        $t_e \leftarrow \frac{fe}{N}$.      /* set generation count after expensive local search */
        if $f(x^*) < f(x^k)$ then
            $x^k \leftarrow x^*$.                               /* replace the old solution */
        end
    end
end
return $x^* = \arg\min_{y \in pop} f(y)$ and $f(x^*)$.
```

where $x = (x_1, x_2, \ldots, x_n)$ denotes a random variable vector and $P_i(x_i)$ is the marginal probability function of x_i.

For a UMDA with a histogram marginal model, the histogram of x_i consists of some bins. Let M denote the number of bins. The mth bin can be described by a tuple $< a_{i,m-1}, a_{i,m}, P_{i,m} >$, where $a_{i,m-1}$ and $a_{i,m}$ are the boundaries of the bin and $P_{i,m}$ denotes the probability that x_i is from the interval $[a_{i,m-1}, a_{i,m})$.

In sampling $P_i(x_i)$, a bin is firstly selected with the probability $P_{i,m}, m = 1, \ldots, M$. A component x_i is then uniformly sampled from $[a_{i,m-1}, a_m)$.

Three types of histogram marginal models are often used for modeling the population distribution in EDAs [43], [44].

1) Equi-width histogram (EWH) model, in which each bin has the same width.
2) Equi-height histogram (EHH) model, in which each bin has the same probability.
3) Cluster-based histogram (CBH) model, in which the points are partitioned into M clusters in each dimension.

Fig. 1(a)–(c) illustrates the basic ideas of EWH, EHH, and CBH, respectively, in the case of $M = 4$.

2) VWH Model: A major shortcoming of the above three models is that they are unable to reach high-precision optimal

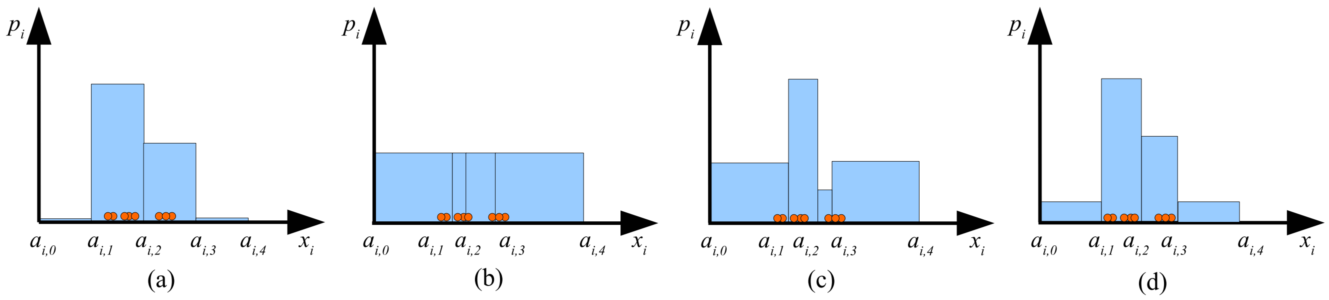


Fig. 1. Illustration of univariate histogram models with continuous decision variables and four bins. (a) EWH. (b) EHH. (c) CBH. (d) VWH.

solutions, particularly when the number of bins, M, is small. To overcome it, we propose a VWH model, which is similar to the adaptive univariate model in [36] and [37]. A VWH model focuses on promising areas and the other areas are assigned a very low probability to avoid premature convergence.

To build the marginal distribution model $P_i(x_i)$ for the VWH model, the search space $[a_i, b_i]$ of the ith variable, x_i, is partitioned into M bins, $[a_{i,m}, a_{i,m+1})$, $m = 0, 1, \ldots, M-2$ and $[a_{i,M-1}, a_{i,M}]$, where $a_{i,0} = a_i$ and $a_{i,M} = b_i$. We let $x^1_{i,\min}$ and $x^2_{i,\min}$ be the first and second minimum values, respectively, and $x^1_{i,\max}$ and $x^2_{i,\max}$ be the first and second maximum values, respectively, of the ith element of the individuals in the current population pop. Set

$$a_{i,1} = \max\left\{x^1_{i,\min} - 0.5\left(x^2_{i,\min} - x^1_{i,\min}\right), a_i\right\}$$
$$a_{i,M-1} = \min\left\{x^1_{i,\max} + 0.5\left(x^1_{i,\max} - x^2_{i,\max}\right), b_i\right\}.$$

The second to $(M-1)$th bins have the same width, that is

$$a_{i,m} - a_{i,m-1} = \frac{1}{M}(a_{i,M-1} - a_{i,1})$$

for $m = 2, \ldots, M-1$. With this partition, all solutions fall into the $M-2$ middle bins which contain promising search regions. No component of the individuals in pop falls into the two end bins.

Let $A_{i,m}$ denote the count of individuals in the mth bin for variable x_i. It is obvious that $A_{i,1} = A_{i,M} = 0$. It is also likely that some bins in the promising areas may contain no solutions. To make sure that each bin has a chance to be searched, we reset them as

$$A_{i,m} = \begin{cases} A_{i,m} + 1 & \text{if } 1 < m < M \\ 0.1 & \text{if } m = 1, M, \text{and } a_{i,m} > a_{i,m-1} \\ 0 & \text{if } m = 1, M, \text{and } a_{i,m} = a_{i,m-1}. \end{cases}$$

If the boundary of the promising area is the same as that of the search space, i.e., $a_{i,0} = a_{i,1} = a_i$ ($a_{i,M} = a_{i,M-1} = b_i$), then set $A_{i,1} = 0$ ($A_{i,M} = 0$); otherwise, set $A_{i,1} = 0.1$ ($A_{i,M} = 0.1$). It should be noted that the count added to the bins in the promising area is higher than that added to the two end bins. The major reason is that we want to focus on the promising area. More discussions are to be found in Section IV-C.

Now, the probability of x_i from the mth bin, $P_{i,m}$ could be approximated as

$$P_{i,m} = \frac{A_{i,m}}{\sum_{j=1}^{M} A_{i,j}}.$$

```
Algorithm 2: $x \leftarrow Sample(P(x))$
Input: the probability model $P(x)$.
Output: a new candidate solution $x$.
for $i \leftarrow 1$ to $n$ do
    Randomly select a bin $m$, according the probability
    $P_{i,j}, j = 1, \ldots, M$.
    Uniformly randomly pick a value $x_i$ from
    $[a_{i,M-1}, a_{i,M}]$ if $m = M$ or from $[a_{i,m-1}, a_{i,m})$ if
    $m < M$.
end
return $x = (x_1, x_2, \ldots, x_n)^T$.
```

Fig. 1(d) illustrates our basic idea in the case of four bins. The VWH model focuses more on the promising search areas especially when the population converges to a small area. In such a case, the space defined by $\prod_{i=1}^{n} [a_{i,1}, a_{i,M-1}]$ might be very small and thus EDA/LS will spend more resources on exploitation.

Each component x_i, $i = 1, \ldots, n$ of solution x is sampled independently and the procedure is shown in Algorithm 2.

C. Cheap LS

To improve the performance of UMDAs, other techniques can be combined with UMDAs. In [32] and [38], classical LS methods are applied to improve some selected solutions. Surrogate models are usually used to estimate the objective values and thus to reduce the number of Fes [45]. In this paper, we propose to improve a solution quality by using a local surrogate model. Since this method does not need any FE and works in a local area, we call it a cheap LS on a surrogate model.

The basic idea of the cheap LS is to replace some components of sampled solutions by those of solutions derived from a local surrogate model with a probability P_c. Fig. 2 illustrates this idea. To generate high-quality solutions, $\lfloor P_b \cdot N \rfloor$ best solutions are used to construct local surrogate models. P_c controls the contribution of the local model. Lines 11–13 in Algorithm 1 shows the details of replacement.

As proposed in [46]–[48], the relation between the objective function and each decision variable z can be approximated by a quadratic model as

$$f = c_1 z^2 + c_2 z + c_3 \tag{2}$$

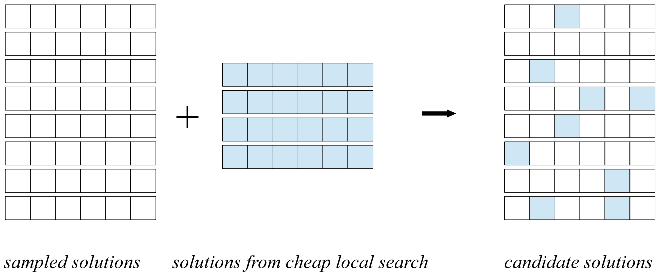


Fig. 2. Illustration of reproduction by combining EDA and local surrogate models in EDA/LS. We assume that the leftmost panel shows the individuals, while the middle panel shows the optimal solutions derived from the local surrogate models. They are combined in a way like crossover to create new offspring.

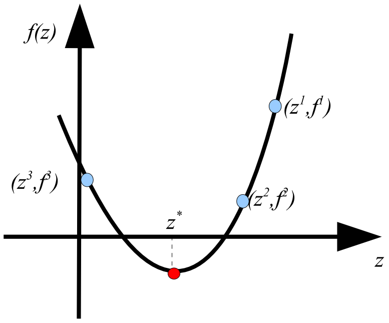


Fig. 3. Illustration of local quadratic approximation to detect local optimal point.

Algorithm 3: $z^* \leftarrow$ *Cheap_ls*$((z^1,f^1), (z^2,f^2), (z^3,f^3))$

```
Input: variable-fitness tuples $(z^1,f^1)$, $(z^2,f^2)$, $(z^3,f^3)$.
Output: minimum point $z^*$.
1 $z^* \leftarrow z^1$.
2 if $|z^1 - z^2| > \varepsilon$ and $|z^2 - z^3| > \varepsilon$ and $|z^3 - z^1| > \varepsilon$ then
3     Calculate $c_1$ and $c_2$ as (3).
4     if $|c_1| > \varepsilon$ then
5         $z^* \leftarrow -\frac{c_2}{2c_1}$.
6     end
7 end
8 return $z^*$.
```

where z is a scalar variable, and c_1, c_2, and c_3 are three parameters. If $c_1 > 0$, its minimum optimal point is located at $z^* = -c_2/2c_1$. Fig. 3 illustrates the idea of local quadratic approximation to find the optimal point.

Given three tuples (z^1,f^1), (z^2,f^2), and (z^3,f^3), it is easy to calculate the parameters in (2) as

$$\begin{aligned} c_1 &= \frac{1}{z^2 - z^3}\left[\frac{f^1 - f^2}{z^1 - z^2} - \frac{f^1 - f^3}{z^1 - z^3}\right] \\ c_2 &= \frac{f^1 - f^2}{z^1 - z^2} - c_1 \frac{z^1 + z^2}{z^1 - z^3}. \end{aligned} \tag{3}$$

In our implementation, the components of three consecutive solutions are used to build the local surrogate model, which is shown in lines 9 and 12 of Algorithm 1. The details of the cheap LS method are shown in Algorithm 3.

In Algorithm 3, $\varepsilon = 1.0 \times 10^{-50}$ is a threshold value to make sure that the points are not too close in line 2. It is also used in line 4. It should be noted that in line 4, we find that the algorithm works better by giving the condition $|c_1| > \varepsilon$ instead of $c_1 > \varepsilon$. The reason might be that a negative c_1 may increase the diversity. A main advantage of the cheap LS is that it actually does not need any FE. However, the local surrogate model is able to capture the characteristics of the local landscape and makes the use of the location information more efficiently.

Algorithm 4: *Converge*(θ, t, t_e)

```
Input:
    1) $\theta$: convergence threshold.
    2) $t$: current generation count.
    3) $t_e$: generation count after the last expensive local
       search.
Output: a Boolean value to indicate whether the current
        population converges or not.
Let $\Delta f \leftarrow \frac{|f^1_{t-50} - f^1_t|}{\max\{|f^1_{t-50}|, |f^1_t|\} + \varepsilon}$, $\Delta x \leftarrow \frac{|c_{t-50} - c_t|}{\max\{c_t, c_{t-50}\} + \varepsilon}$.
if $t > t_e + 50$ and $\min\{\Delta f, \Delta x\} < \theta$ then
    return true.
else
    return false.
end
```

D. Expensive LS

It is well known that EAs are not very efficient in refining promising solutions especially in the later stages. Therefore, it is reasonable to use other LS methods when the evolutionary search has reached its convergence stage. Here, we call it an expensive LS to distinguish it from the cheap LS.

The challenge is how to determine whether or not an algorithm has converged. We propose an *ad hoc* procedure for this purpose. Its details are given in Algorithm 4.

In Algorithm 4, Δf denotes the decreasing ratio of the best fitness value over the last 50 generations, where $f^1_t = \min_{x \in \text{pop}_t} f(x)$ is the best objective at generation t, and Δx denotes the changing ratio of the area covered by the population over the last 50 generations, where $c_t = \frac{1}{n}\sum_{i=1}^{n}(\max_{x\in \text{pop}} x_i - \min_{x \in \text{pop}} x_i)$. $\varepsilon = 1.0 \times 10^{-50}$ is a small value. A convergence happens when either Δf or Δx is below a given threshold θ. Furthermore, we require there are at least 50 generations between two expensive searches.

The parameter θ is usually problem-dependent. Since the change ratios in both the decision space and the objective space are considered over the last 50 generations, the problem-dependency could be reduced. It should be noted that other convergence conditions could also be applied here.

Once the population has been considered convergent, we use a modified Powell method [39], [40], denoted as expensive_ls in line 23 of Algorithm 1, to improve one of the best solutions found so far. The Powell method is a classical optimization method based on trust region. It does not use any gradient information. Instead, the Hessian matrix is approximated by interpolation of carefully chosen solutions and minimization of

Frobenius norm. It has been demonstrated to be very capable in locating local optima.

In $\{x^*, fl\} \leftarrow$ expensive_ls(x, fm), x is the start point, fm is the maximum FEs allowed, x^* is the optimal point found by the Powell method, and fl is the actual number of FEs used in the search. In the original Powell method, it terminates if $2(f_{\text{iter}-1} - f_{\text{iter}}) \leq f_{\text{tol}}(|f_{\text{iter}-1}| + |f_{\text{iter}}| + \varepsilon)$ where f_{iter} and $f_{\text{iter}-1}$ denote the function value of the current and last iterations respectively, $f_{\text{tol}} = 1.0 \times 10^{-10}$ is the tolerance, and $\varepsilon = 1.0 \times 10^{-50}$ is a small value. Our modifications are: 1) only the feasible solutions are accepted and recorded and 2) the Powell method terminates if the original stop condition is satisfied or the number of FEs reaches the given maximum number fm. The details of the Powell method and the source codes could be found in [40].

III. Algorithms For Comparison

This section briefly introduces two state-of-the-art algorithms, Jingqiao and Arthur differential evolution (JADE) [49] and DE/DEA [38], and two EDAs based on univariate histogram marginal models for comparison.

A. JADE

JADE [49] is an improved version of differential evolution (DE) [50] with a new mutation strategy "DE/current-to-*p*best," an external archive and an adaptive parameter control scheme. It has been shown in [49] that JADE is better than jDE [51], SaDE [52], the classical DE/rand/1/bin, a particle swarm optimization algorithm [53], and two variate JADEs on some test instances. JADE is one of the best EAs for global optimization and this is why we choose it for comparison. The details of JADE could be found in [49].

B. DE/EDA

DE/EDA [38], uses a similar idea in offspring reproduction as EDA/LS. It hybridizes both the EDA sampling and DE reproduction strategies for generating new trial solutions. The offspring is partly sampled from a fully-factorized Gaussian model and partly from DE crossover. The experimental results on seven test instances with both 5 and 10 variable dimensions shown that DE/EDA is better than pure EDA and DE algorithms. The details of DE/EDA could be found in [38].

C. EDA/EWH and EDA/EHH

EDA/EWH and EDA/EHH, are two algorithms using only probabilistic models for generating new trial solutions. EDA/EWH uses the EWH model as shown in Fig. 1(a), and EDA/EHH uses the EHH model as shown in Fig. 1(b). The initialization, the stop condition, and the selection operators are the same as in EDA/LS.

IV. Performance on YLL Test Suite

A. Test Instances

The performance of EDA/LS is assessed by the first 13 test instances from the YLL test suite [41]. All the test instances have a global minimum objective value 0. $f1$–$f4$ are unimodal functions. $f5$ is a unimodal function when $n = 2$ and $n = 3$, and may have many local optimal solutions when $n > 3$. $f6$ is a step function. $f7$ is a function with white noise. $f8$–$f13$ are multimodal functions with many local optimal solutions. The details of YLL test suite are referred to [41].

B. Parameter Settings

As discussed in [24], pure EDAs may need large population size. To have a fair comparison, we have conducted some experiments to optimize some control parameters of the comparison algorithms.[1] The parameter settings of the experiments are as follows.

1) The variable dimensions for all test instances $n = 30$; all algorithms are executed independently for 50 runs and stopped after 300 000 FEs.
2) *EDA/LS:* The number of bins used in model building $M = 15$; the population size $N = 150$; the percentage of best solutions used in the local surrogate model $P_b = 0.2$; the probability to use location information $P_c = 0.2$; and the convergence threshold $\theta = 0.1$.
3) *JADE:* The parameters $N = 100$, $p = 0.05$, $c = 0.1$, $F = 0.5$, and $CR = 0.9$ as suggested in [49].
4) *DE/EDA:* The parameters $N = 150$, $F = 0.5$, and $\theta = 0.9$, which are based on the experimental results in [38].
5) *EDA/EWH:* The population size $N = 2000$; the number of bins $M = 100$.
6) *EDA/EHH:* The population size $N = 1000$; the number of bins $M = 100$.

All the algorithms are implemented in MATLAB R2013b[2] and executed in Lenovo Thinkpad W530 with i7-3470 QM CPU @ 2.70 GHz, 8.00 GB RAM, and Windows 7.

C. Results and Analysis

Table I summarizes the mean and standard deviation of the results obtained by the five algorithms after 3.0×10^5 FEs over 50 independent runs. The Wilcoxon rank sum test is applied to statically compare the function values obtained by EDA/LS and that of another algorithm. The results of Wilcoxon rank sum test are shown in Table I. Table II presents the average evaluations and number of success runs required to achieve the goal of $f < 1.0 \times 10^{-14}$. Fig. 4 shows that the mean function values obtained by the five algorithms versus the FEs. The analyses of the statistical results are as follows.

1) EDA/LS Versus EDA/EWH and EDA/EHH: Table I shows that, according to the Wilcoxon rank sum test on the solutions found by these algorithms, EDA/EWH performs worse than EDA/LS on all the 13 test instances, and EDA/EHH performs similar to EDA/LS on $f6$ and worse than EDA/LS on the other instances. Table II also indicates that only EDA/EHH can obtain reasonably good results on $f6$ in all runs.

It is strange that both EDA/EWH and EDA/EHH fail even on the simplest instance $f1$. Fig. 4 shows that, the average

[1] EDA/EWH and EDA/EHH are tested with $M = 10, 25, 50$, and 100, and $N = 250, 500, 750, 1000, 2000$, and 3000.

[2] The source code of JADE is from the authors. DE/EDA, EDA/EWH, and EDA/EHH were implemented by ourselves. The source codes are available from www.cs.ecnu.edu.cn/~amzhou

TABLE I

STATISTICAL RESULTS (MEAN ± STD) FOR EDA/LS, JADE, DE/EDA, EDA/EWH, AND EDA/EHH ON $f1$–$f13$. "+," "−," OR "∼" IN PARENTHESES INDICATES THE FUNCTION VALUE IS SMALLER THAN, GREATER THAN, OR SIMILAR TO THAT OBTAINED BY EDA/LS AT 95% SIGNIFICANCE LEVEL BY A WILCOXON RANK SUM TEST

	EDA/LS	JADE	DE/EDA	EDA/EWH	EDA/EHH
$f1$	4.05e-130±3.31e-130	**1.44e-135**±7.41e-135(+)	7.46e-70±8.85e-70(-)	2.98e+01±3.40e+00(-)	1.48e-02±3.17e-02(-)
$f2$	**9.12e-65**±3.80e-65	1.14e-30±5.00e-30(-)	1.40e-33±1.18e-33(-)	1.76e+00±9.98e-02(-)	7.48e-04±9.02e-04(-)
$f3$	**1.11e-35**±2.94e-35	4.93e-35±1.46e-34(-)	2.42e-15±4.39e-15(-)	2.02e+04±2.75e+03(-)	6.66e+03±1.30e+03(-)
$f4$	**1.02e-37**±7.22e-37	1.62e-14±3.46e-14(-)	9.80e-08±2.27e-07(-)	1.81e+01±8.99e-01(-)	2.83e+00±5.97e-01(-)
$f5$	**3.26e-29**±2.83e-29	7.97e-02±5.64e-01(+)	2.39e-01±9.56e-01(-)	2.79e+03±6.11e+02(-)	1.42e+02±3.54e+01(-)
$f6$	**0.00e+00**±0.00e+00	**0.00e+00**±0.00e+00(∼)	**0.00e+00**±0.00e+00(∼)	3.25e+01±5.07e+00(-)	**0.00e+00**±0.00e+00(∼)
$f7$	2.41e-03±6.69e-04	**6.83e-04**±3.24e-04(+)	2.03e-03±5.15e-04(+)	5.17e-02±8.12e-03(-)	1.12e-02±1.90e-03(-)
$f8$	**0.00e+00**±0.00e+00	**0.00e+00**±0.00e+00(∼)	4.74e+02±2.11e+02(-)	4.73e+01±5.09e+00(-)	6.20e-03±1.53e-02(-)
$f9$	1.99e-02±1.41e-01	**0.00e+00**±0.00e+00(∼)	5.88e+01±4.58e+01(-)	1.10e+01±6.57e-01(-)	1.05e-02±2.55e-02(-)
$f10$	**4.44e-15**±0.00e+00	**4.44e-15**±0.00e+00(∼)	4.65e-15±8.52e-16(∼)	3.39e+00±1.21e-01(-)	1.08e-02±8.39e-03(-)
$f11$	**0.00e+00**±0.00e+00	**0.00e+00**±0.00e+00(∼)	8.38e-04±3.01e-03(-)	1.27e+00±3.62e-02(-)	1.15e-02±1.75e-02(-)
$f12$	**1.57e-32**±5.53e-48	**1.57e-32**±5.53e-48(∼)	2.07e-03±1.47e-02(∼)	2.55e+00±5.40e-01(-)	2.53e-05±4.40e-05(-)
$f13$	**1.35e-32**±1.11e-47	**1.35e-32**±1.11e-47(∼)	4.39e-04±2.17e-03(∼)	1.95e+01±2.38e+00(-)	2.96e-03±4.05e-03(-)

TABLE II

AVERAGE FEs AND NUMBER OF SUCCESSFUL RUNS TO ACHIEVE $f < 1.0 \times 10^{-14}$ ON $f1$–$f13$ FOR EDA/LS, JADE, DE/EDA, EDA/EWH, AND EDA/EHH. FIRST VALUE IS AVERAGE FEs IN TERMS OF 10^5 AND THE BRACKETED VALUE IS THE NUMBER OF SUCCESSFUL RUNS

	EDA/LS	JADE	DE/EDA	EDA/EWH	EDA/EHH
$f1$	0.40(50)	0.42(50)	0.75(50)	NA(0)	NA(0)
$f2$	0.73(50)	0.82(50)	1.38(50)	NA(0)	NA(0)
$f3$	1.15(50)	1.39(50)	2.82(47)	NA(0)	NA(0)
$f4$	1.10(50)	2.77(36)	NA(0)	NA(0)	NA(0)
$f5$	0.68(50)	1.75(49)	2.73(46)	NA(0)	NA(0)
$f6$	0.10(50)	0.11(50)	0.19(50)	NA(0)	1.04(50)
$f7$	NA(0)	NA(0)	NA(0)	NA(0)	NA(0)
$f8$	0.68(50)	1.57(50)	2.37(1)	NA(0)	NA(0)
$f9$	1.70(49)	1.73(50)	NA(0)	NA(0)	NA(0)
$f10$	0.70(50)	0.73(50)	1.30(50)	NA(0)	NA(0)
$f11$	0.42(50)	0.47(50)	0.76(46)	NA(0)	NA(0)
$f12$	0.37(50)	0.42(50)	0.70(49)	NA(0)	NA(0)
$f13$	0.39(50)	0.44(50)	0.74(48)	NA(0)	NA(0)

function values of best solutions do not decrease much. To figure out why, we record the width of the bin which contained the optimal value 0 and the probability associated with the bin for variable x_1 by EDA/EWH, EDA/EHH, and EDA/LS on $f1$. Since $f1$ is unimodal and can be fully factorized, the behavior on other variables should be the same. Fig. 5 shows the statistical results.

From the figure, it can be observed that in the search process of EDA/EWH, the average probability with the bin increases to 0.4; for EDA/EHH, the width of the bin does not change much. The figure also explains why EDA/EWH and EDA/EHH fail on $f1$. For EDA/EWH, although the probability with the bin is high, the width of each bin is 2.00 and thus the chance of sampling an element very close to 0 is still very low. For EDA/EHH, the average width of the bin is 4.00 after 30 000 FEs and thus it is very unlikely to sample a value very close to 0. For the same reason, neither EDA/EWH nor EDA/EHH is able to sample high-quality solutions for all the other 29 variables. In EDA/LS, although the average probability with the bin is around 0.15, the width drops to near 10^{-66} after 300 000 FEs.

Let us make more comments on the search behaviors of the three univariate histogram models. As illustrated in Fig. 6, the width of the bin in EWH that contains the optimum is fixed. Although its probability is high, it is not easy to sample a solution of high precision. In EHH, the probability of each bin is fixed and some bins might be narrow. However, it is not likely to reduce the width of the bin that contains the optimum because it may sample high quality candidates for each variable independently but may not sample high quality candidates for all the variables. Therefore, it is difficult to sample high-precision solutions. In VWH, since the bins in the promising area is becoming narrow as the search goes, the bin that contains the optimum is also becoming narrow and it is very likely to sample high-quality solutions as the algorithm runs. To sum up, the width of the bins in promising area is crucial. Neither EWH nor EHH can guarantee to reduce the width of the bin that contains the optimum, but VWH can.

The bad performance of EDA/EWH and EDA/EHH on the other instances could be explained in the same way. It is evident that the VWH model is more capable than the other two models to obtain the optimal element with very high precision.

2) EDA/LS Versus DE/EDA: Both EDA/LS and DE/EDA combine EDA model techniques with other techniques for offspring reproduction. Table I shows that for all the instances expect $f6$, $f10$, $f12$, and $f13$, DE/EDA has performed worse

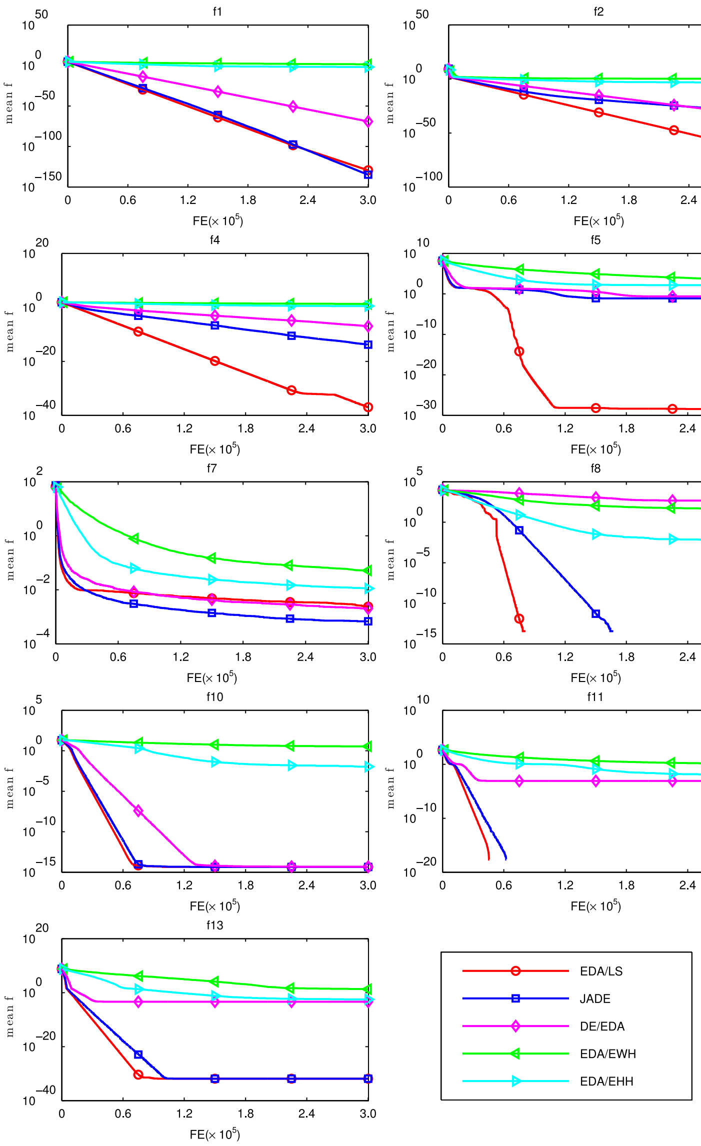
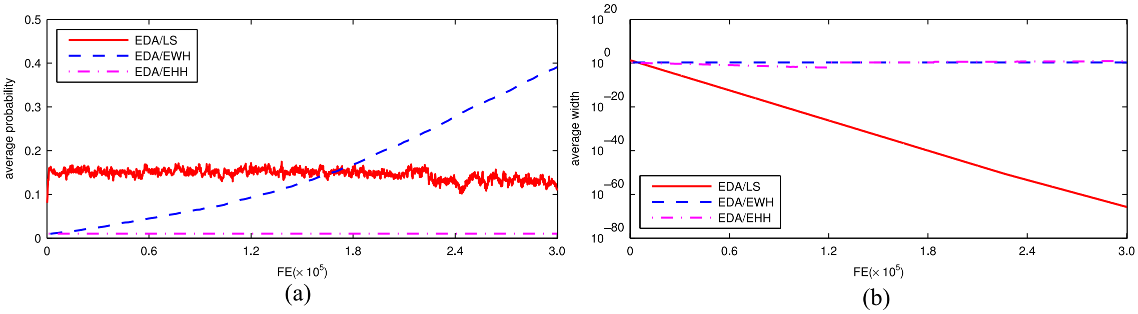


Fig. 4. Mean function value of the best solutions obtained by the five methods versus FE on $f1$–$f13$.



(a)

(b)

Fig. 5. Statistical results of the bin covering the optimal solution 0 of the variable x_1 obtained by EDA/LS, EDA/EWH, and EDA/EHH on $f1$ versus FE. (a) Average probability value. (b) Average bin width.

than EDA/LS based on the Wilcoxon rank sum test. However, the mean objective values indicate that on $f12$ and $f13$, DE/EDA is worse than EDA/LS. As shown in Table II, one reason should be that DE/EDA has failed in one run on $f12$ and in two runs on $f13$. Although it has performed not as good as EDA/LS, DE/EDA can finally obtain high-quality solutions in most of runs on $f1-f3$, $f5$, and $f11$.

From Fig. 4 and Table II, it is clear that DE/EDA converges slower than EDA/LS. The slow convergence of DE/EDA might due to its univariate Gaussian models for the following two reasons: 1) in sampling the Gaussian model, the offspring is possibly far from the parents and 2) the probability distribution of the population is actually not a Gaussian and thus a Gaussian model may mislead the search. However, it can be

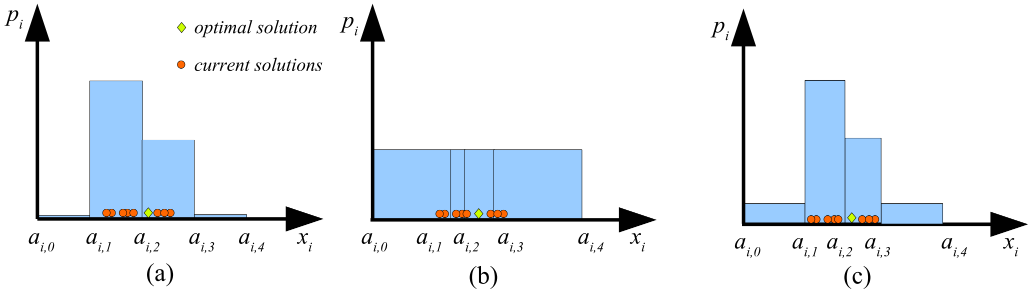


Fig. 6. Illustration of the search behaviors of univariate histogram models. (a) EWH. (b) EHH. (c) VWH.

seen from the figure that these two shortcomings have been overcome in our univariate histogram model. Possible reasons are that the proposed histogram model adaptively changes to suit the drift of promising areas and the LS may also help drive, not mislead, the search to promising areas.

3) EDA/LS Versus JADE: From Fig. 4, it is evident that on average, EDA/LS and JADE are comparable on most test instances. The Wilcoxon rank sum test results in Table I shows that on $f1$, $f5$, and $f7$, JADE is better than EDA/LS, on $f2$, $f3$, and $f4$, EDA/LS performs better, and on all other seven instances, the final solutions obtained by both EDA/LS and JADE are quite similar.

$f7$ is a function with white noises, i.e., uniformly distributed in [0, 1). The challenge is that the objective value of a solution is not deterministic. To obtain an objective value of high precision, a solution needs to be evaluated for many times. Since the algorithms do not consider this issue, neither EDA/LS nor JADE can produce very good solutions. For optimization in noisy or dynamic environments, interested readers are referred to [54].

It should be noticed that $f5$, and $f8$–$f13$ are multimodal functions. But both EDA/LS and JADE could find high-quality solutions. The global optimum on $f10$, $f12$, and $f13$ are 0. However, both EDA/LS and JADE converge to almost the same local optimum. It should be because of their landscape characteristics, which will be a future research topic for us to explore.

A major advantage of EDA/LS over JADE is that EDA/LS converges faster than JADE on all test instances except $f7$. Taking $f10$–$f13$ as examples, Table II shows that the FEs used by EDA/LS were 63.3%, 60.2%, 58.3%, 59.1% of those used by JADE to achieve the goal. It can also confirmed by Fig. 4.

4) Summary: The experimental results in this section have demonstrated that the proposed VWH model is more capable than the EWH model and the EHH model in locating high-quality solutions. Moreover, DE/EDA could get good results for some test instances. JADE and EDA/LS are comparable in terms of solution quality. However, EDA/LS is faster than JADE on most instances.

D. Roles of EDA/LS Components

The main components of EDA/LS include the VWH model, the cheap LS and the expensive LS. This section investigates the effects of these components on the performance of EDA/LS.

For convenience, we use m, c, and e to denote the VWH model component, the cheap LS and the expensive LS, respectively. We test the following combinations of these components.

1) *EDA/LS$_m$:* Pure VWH model-based approach without cheap or expensive LS.
2) *EDA/LS$_{m,c}$:* VWH model with cheap LS and without expensive LS.
3) *EDA/LS$_{m,e}$:* VWH model with expensive search and without cheap LS.
4) *EDA/LS$_{m,c,e}$:* VWH model with all the components, which is actually the EDA/LS itself.

For a fair comparison, we replace the VWH model by uniformly random sampling from the search space. Denote this approach as RAND. The variants of RAND search used in our experiments are RAND$_m$, RAND$_{m,c}$, RAND$_{m,e}$, and RAND$_{m,c,e}$. Their meanings are the same as the notations for EDA/LS except that the VWH model is replaced by random sampling. We also use the Powell method, denoted as POWELL, with a restart strategy in comparison. The test instances $f1$–$f13$ are used. The algorithm parameters are the same as in Section IV-B.

The average number of calls to the expensive LS and the FEs used by the expensive LS in the algorithms with different components are shown in Table III. The average FEs and number of successful runs to achieve $f < 1.0 \times 10^{-14}$ are shown in Table IV. In the following, we analyse the three major components one-by-one.

1) VWH Model: Table IV shows that RAND$_m$ fails on all the instances, while EDA/LS$_m$ fails on $f3$, $f5$, and $f7$, and works very well on the other instances. Comparing EDA/LS$_{m,c,e}$ and RAND$_{m,c,e}$, although RAND$_{m,c,e}$ converges faster than EDA/LS$_{m,c,e}$ on $f1$, $f2$, $f3$, $f5$, $f6$, $f12$, and $f13$, it can obtain good solutions only on three runs for $f4$ and fails on $f7$–$f10$, while EDA/LS$_{m,c,e}$ could find good solutions in almost all the 50 runs on all the instances except on $f7$. These results indicate that the VWH model is more stable than the random sampling strategy.

2) Cheap LS: Both EDA/LS$_m$ and EDA/LS$_{m,c}$ fail on $f3$, $f5$, and $f7$. They work well on all the other instances, However, EDA/LS$_{m,c}$ achieves the goals with much less FEs than EDA/LS$_m$ on these instances except on $f9$. The comparison between EDA/LS$_{m,e}$ and EDA/LS$_{m,c,e}$ also shows that except on $f3$ and $f9$, the computation cost can be reduced by using the local surrogate model. This suggests that, as discussed in Section II-C, the cheap LS can help to refine the solutions in local areas successfully.

TABLE III
AVERAGE CALLS AND THE FES USED BY THE EXPENSIVE LS IN THE ALGORITHMS WITH DIFFERENT COMPONENTS ON $f1$–$f13$. FIRST VALUE IS THE AVERAGE NUMBER OF CALLS AND THE SECOND IS THE AVERAGE NUMBER OF FUNCTION EVOLUTIONS IN TERMS OF 10^5

	EDA/LS$_m$	EDA/LS$_{m,c}$	EDA/LS$_{m,e}$	EDA/LS$_{m,c,e}$	RAND$_m$	RAND$_{m,c}$	RAND$_{m,e}$	RAND$_{m,c,e}$	POWELL
$f1$	(0.00,0.00)	(0.00,0.00)	(0.00,0.00)	(0.00,0.00)	(0.00,0.00)	(0.00,0.00)	(21.04,1.36)	(20.98,1.36)	(36.34,3.00)
$f2$	(0.00,0.00)	(0.00,0.00)	(0.00,0.00)	(0.00,0.00)	(0.00,0.00)	(0.00,0.00)	(5.32,2.57)	(5.34,2.57)	(1.20,3.00)
$f3$	(0.00,0.00)	(0.00,0.00)	(5.74,2.53)	(5.02,1.89)	(0.00,0.00)	(0.00,0.00)	(5.64,2.54)	(5.70,2.54)	(4.18,3.00)
$f4$	(0.00,0.00)	(0.00,0.00)	(0.00,0.00)	(0.00,0.00)	(0.00,0.00)	(0.00,0.00)	(5.84,2.52)	(6.08,2.51)	(1.98,3.00)
$f5$	(0.00,0.00)	(0.00,0.00)	(6.28,2.27)	(6.14,2.37)	(0.00,0.00)	(0.00,0.00)	(6.54,2.47)	(6.34,2.48)	(4.84,3.00)
$f6$	(0.00,0.00)	(0.00,0.00)	(36.00,0.05)	(37.00,0.06)	(0.00,0.00)	(0.00,0.00)	(36.10,0.19)	(36.20,0.18)	(369.04,3.00)
$f7$	(0.00,0.00)	(0.00,0.00)	(7.20,2.24)	(7.40,2.25)	(0.00,0.00)	(0.00,0.00)	(5.54,2.54)	(5.58,2.54)	(3.54,3.00)
$f8$	(0.00,0.00)	(0.00,0.00)	(33.14,0.32)	(31.20,0.20)	(0.00,0.00)	(0.00,0.00)	(31.78,0.54)	(31.46,0.56)	(165.22,3.00)
$f9$	(0.00,0.00)	(0.00,0.00)	(29.52,0.54)	(30.22,0.55)	(0.00,0.00)	(0.00,0.00)	(30.12,0.66)	(29.48,0.71)	(94.52,3.00)
$f10$	(0.00,0.00)	(0.00,0.00)	(22.00,0.26)	(25.94,0.31)	(0.00,0.00)	(0.00,0.00)	(21.64,1.31)	(18.80,1.53)	(41.88,3.00)
$f11$	(0.00,0.00)	(0.00,0.00)	(28.76,0.13)	(31.00,0.14)	(0.00,0.00)	(0.00,0.00)	(32.12,0.50)	(32.00,0.52)	(131.62,3.00)
$f12$	(0.00,0.00)	(0.00,0.00)	(21.04,0.21)	(25.78,0.24)	(0.00,0.00)	(0.00,0.00)	(27.80,0.84)	(27.94,0.83)	(73.02,3.00)
$f13$	(0.00,0.00)	(0.00,0.00)	(18.90,0.32)	(22.90,0.42)	(0.00,0.00)	(0.00,0.00)	(24.72,1.08)	(23.98,1.13)	(54.74,3.00)

TABLE IV
AVERAGE FES AND NUMBER OF SUCCESSFUL RUNS TO ACHIEVE $f < 1.0 \times 10^{-14}$ ON $f1$–$f13$ BY THE ALGORITHMS WITH DIFFERENT COMPONENTS. FIRST VALUE IS THE AVERAGE NUMBER OF FES IN TERMS OF 10^5 AND THE BRACKETED VALUE IS THE NUMBER OF SUCCESSFUL RUNS

	EDA/LS$_m$	EDA/LS$_{m,c}$	EDA/LS$_{m,e}$	EDA/LS$_{m,c,e}$	RAND$_m$	RAND$_{m,c}$	RAND$_{m,e}$	RAND$_{m,c,e}$	POWELL
$f1$	0.59(50)	0.40(50)	0.60(50)	0.40(50)	NA(0)	NA(0)	0.08(50)	0.08(50)	0.00(50)
$f2$	1.00(50)	0.73(50)	1.00(50)	0.73(50)	NA(0)	NA(0)	0.28(50)	0.30(50)	0.19(50)
$f3$	NA(0)	NA(0)	0.48(50)	1.15(50)	NA(0)	NA(0)	0.50(50)	0.48(50)	0.43(50)
$f4$	2.39(50)	1.10(50)	2.38(50)	1.09(50)	NA(0)	NA(0)	2.26(1)	1.31(3)	2.01(2)
$f5$	NA(0)	NA(0)	0.73(50)	0.68(50)	NA(0)	NA(0)	0.69(50)	0.64(50)	0.59(50)
$f6$	0.15(50)	0.10(50)	0.15(50)	0.10(50)	NA(0)	NA(0)	0.08(50)	0.08(50)	0.01(50)
$f7$	NA(0)	NA(0)	NA(0)	NA(0)	NA(0)	NA(0)	NA(0)	NA(0)	NA(0)
$f8$	0.86(50)	0.63(50)	0.99(38)	0.68(50)	NA(0)	NA(0)	NA(0)	NA(0)	NA(0)
$f9$	1.73(50)	2.41(47)	1.56(50)	1.70(49)	NA(0)	NA(0)	NA(0)	NA(0)	NA(0)
$f10$	1.02(50)	0.69(50)	1.02(50)	0.70(50)	NA(0)	NA(0)	NA(0)	NA(0)	NA(0)
$f11$	0.61(50)	0.42(50)	0.61(50)	0.42(50)	NA(0)	NA(0)	0.89(48)	0.59(48)	0.16(50)
$f12$	0.55(50)	0.37(50)	0.55(50)	0.37(50)	NA(0)	NA(0)	0.28(50)	0.34(50)	0.08(50)
$f13$	0.58(50)	0.39(50)	0.58(50)	0.39(50)	NA(0)	NA(0)	0.15(50)	0.18(50)	0.05(50)

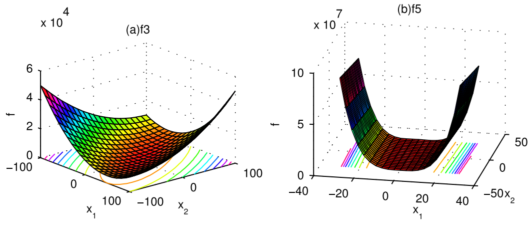


Fig. 7. Illustration of (a) $f3$ and (b) $f5$ in the case of $n = 2$.

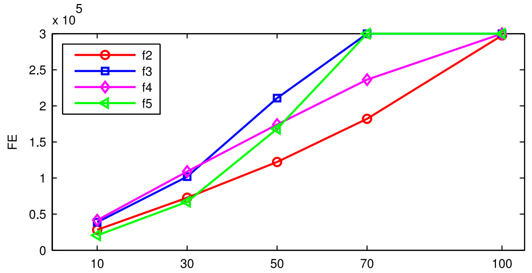


Fig. 8. Mean FEs required to achieve the goal $f < 10^{-14}$ for EDA/LS on $f2$–$f5$ with different n.

3) Expensive LS: It is evident from Table IV that by using the expensive LS, both EDA/LS$_{m,e}$ and EDA/LS$_{m,c,e}$ can solve all the instances except $f7$, while without it, EDA/LS$_m$ and EDA/LS$_{m,c}$ fail not only on $f7$ but also on $f3$ and $f5$. The Powell method also improves the search quality of the random sampling strategy. Table III shows that on $f1$, $f2$, and $f4$, the number of calls to the Powell method was 0 in both EDA/LS$_{m,e}$ and EDA/LS$_{m,c,e}$. This suggests that the

TABLE V
STATISTICAL RESULTS (MEAN ± STD) FOR EDA/LS ON $f2$–$f5$ WITH VARIABLE DIMENSION OF $n = 10, 30, 50, 70, 100$

n	$f2$	$f3$	$f4$	$f5$
10	2.94e-155±2.88e-155	1.04e-69±7.35e-69	2.66e-68±1.66e-67	9.89e-32±6.99e-31
30	1.02e-64±8.87e-65	1.79e-34±1.16e-33	2.18e-42±1.33e-42	2.52e-29±2.83e-29
50	3.21e-38±1.56e-38	1.34e+01±4.78e+01	1.47e-14±1.04e-13	1.47e-04±1.04e-03
70	3.01e-25±1.34e-25	4.84e+02±4.62e+02	8.48e-13±6.00e-12	1.17e+01±3.92e+00
100	7.95e-15±2.65e-15	4.27e+03±2.43e+03	1.81e-08±1.23e-07	6.34e+01±1.36e+01

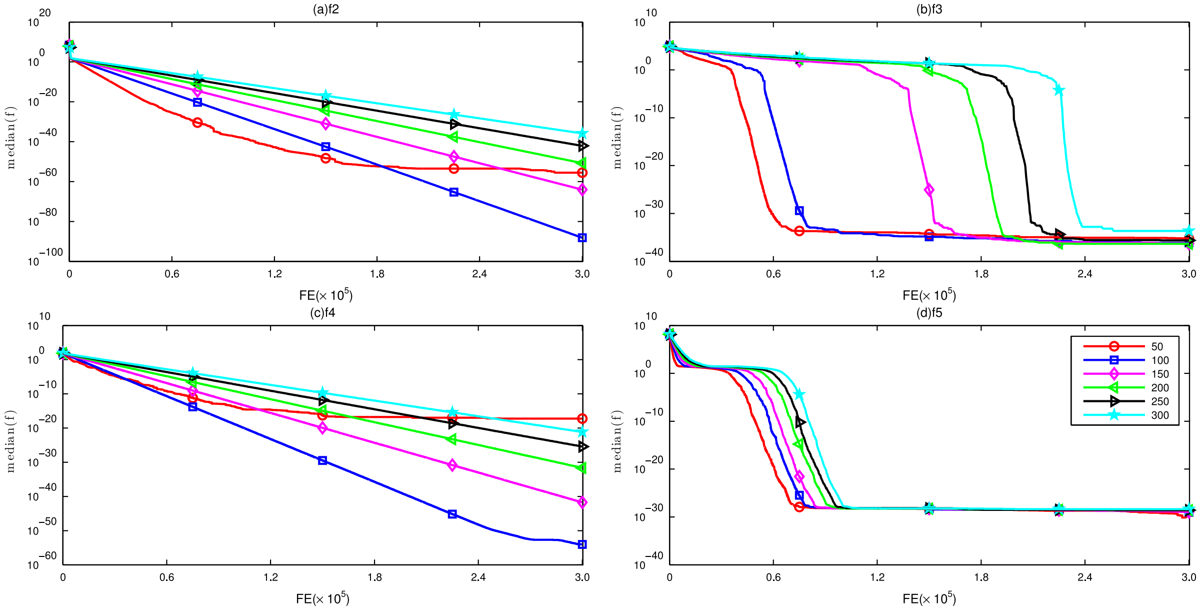


Fig. 9. Median objective function values of the best solutions obtained by EDA/LS with different N versus FE on $f2$–$f5$.

convergence test does not return true value on these three instances. The reason should be that without the Powell method, EDA/LS$_m$ and EDA/LS$_{m,c}$ could work well thus the Powell method is not necessary. Let us further investigate why the Powell method is necessary on $f3$ and $f5$. Fig. 7 illustrates the two functions in the case of $n = 2$ and it shows that the objective surfaces are very "flat" in the sense that the objective function values do not change much in some areas near the optimum. Without the Powell method, EDA/LS$_m$ and EDA/LS$_{m,c}$ may spend a lot of effort in these flat areas. The Powell method could speed up the search. It should be noted from Table IV that, a pure Powell method with a restart strategy also works well on $f1$, $f2$, $f3$, $f5$, $f6$, $f11$, $f12$, and $f13$ but fails on the other five instances. The reason might be that it may need an initial solution that is close to the optimum.

4) Summary: Overall, we can conclude that for easy instances like $f1$, $f2$, and $f4$, the VWH model with or without cheap and expensive LS are able to perform well. However, for hard instances like $f3$ and $f5$, the cheap and expensive LS can work collaboratively to achieve good results.

E. Sensitivity to Variable Dimension n

In previous sections, EDA/LS has been tested on instances with $n = 30$. It would be interesting to see whether EDA/LS is sensitive to the number of variables. In this section, we apply EDA/LS to $f2$–$f5$ with $n = 10, 30, 50, 70$, and 100. The algorithm parameters are the same as in Section IV-B.

Fig. 8 shows the average number of FEs required to achieve the goal $f < 1.0^{-14}$. The final results obtained by EDA/LS after 300 000 FEs are presented in Table V.

From Fig. 8, it can be seen that the performance of EDA/LS decreases as n increases. However, its performance also depends on problems. It is shown in Table V that on $f2$ and $f4$, EDA/LS could find high-quality solutions even n is high. However, on $f3$ and $f5$ with $n \geq 50$, EDA/LS performs not as well as on $f2$ and $f4$. A reason could be that, as discussed in the previous section, EDA/LS may need more efforts to do expensive LS.

F. Sensitivity to Algorithm Parameters

In this section, we investigate the sensitivity of EDA/LS to the population size N, the number of bins M, the percentage of best solutions used in the LS P_b, the probability to use location information P_c, and the convergence threshold θ. The instances $f2$–$f5$ are used in the experiments. To test each parameter, all other parameters are fixed the same as in Section IV-B. It should be noticed that in an EA, the components (with different parameter settings) interact one another and it is hard to find a universal configuration [3].

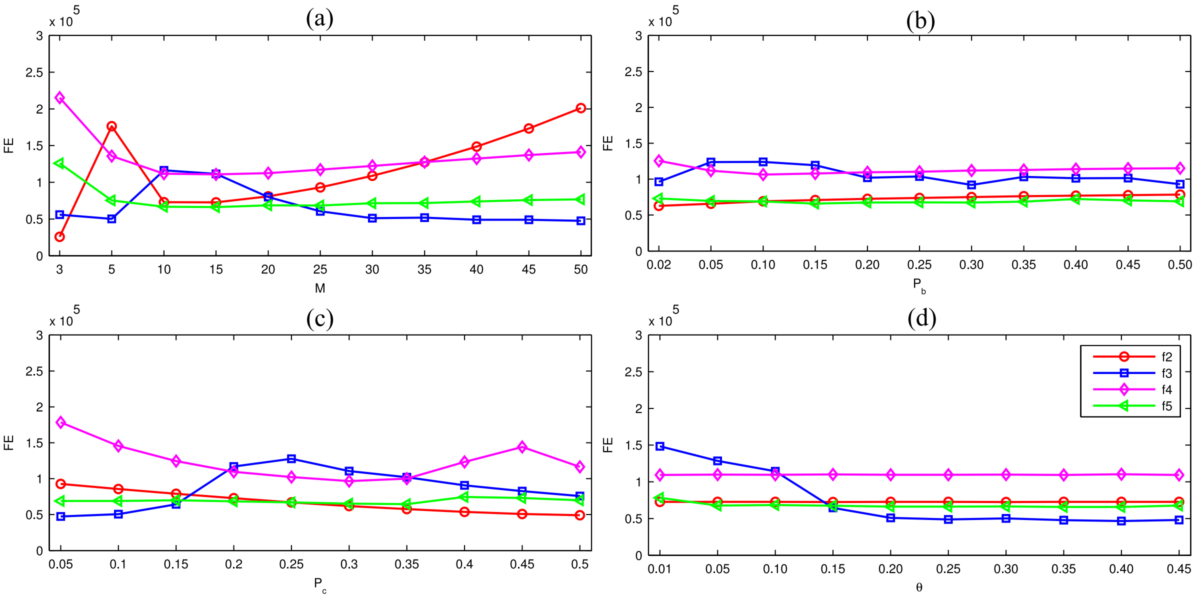


Fig. 10. Mean FEs used to achieve the goal $f < 10^{-14}$ for EDA/LS with different control parameters on $f2$–$f5$. (a) M. (b) P_b. (c) P_c. (d) θ.

1) Sensitivity to N: EDA/LS with population size $N = 50, 100, 150, 200, 250$, and 300 are run on the test instances. Fig. 9 shows the median function values against the FEs for these population settings.

The results in Fig. 9 shows that EDA/LS could achieve high-quality solutions on all the four instances with all the six different population settings. On $f2$ and $f4$, EDA/LS with $N = 50$ converges quickly in its early stage and then gets stuck. For EDA/LS with $100 \leq N \leq 300$, the solution quality becomes worse as the population size increases. On $f3$ and $f5$, EDA/LS with all the six population sizes achieves similar final results, but the smaller the population size is, the faster the convergence is. These results suggest that EDA/LS with too small population size like $N = 50$ may be unstable, and EDA/LS with too large population size may work but its performance becomes worse. EDA/LS with population size around 100–150 may lead to good balance between the solution quality and the convergence speed.

With the given number of FEs, the population size of an EDA should be carefully set. If the population size is too small, it may be hard to build a model of good quality. On the other hand, if the population size is too large, the number of generations will be reduced and thus the algorithm may not be able to find optimal solutions. How to set the population size is worthwhile further investigating.

2) Sensitivity to M: The number of bins is a key parameter in building the probabilistic model in EDA/LS. In this section, EDA/LS with $M = 3, 5, 10, \ldots, 50$ is tested. Fig. 10(a) presents the average number of FEs used to achieve the goal $f < 10^{-14}$.

From the figure, it is clear that on $f4$ and $f5$, EDA/LS with different M values achieve stable results, and on $f2$ and $f3$, it is unstable. On $f2$, as the value of M increases, the performance of EDA/LS first becomes worse, then better, and finally worse again. On $f3$, the performance of EDA/LS first becomes better, then worse, and finally better. However, EDA/LS with different M values can achieve the goal $f < 10^{-14}$ within about 2.0×10^5 FEs. The results suggest that EDA/LS is not very sensitive to the number of bins.

TABLE VI
AVERAGE CPU TIME (SECONDS) USED BY EDA/LS, JADE, DE/EDA, EDA/EWH, AND EDA/EHH ON $f1$–$f13$

	EDA/LS	JADE	DE/EDA	EDA/EWH	EDA/EHH
$f1$	2.67	1.99	2.23	1.59	1.90
$f2$	2.79	2.02	2.30	1.62	1.91
$f3$	8.03	1.84	2.23	1.62	1.89
$f4$	2.74	1.88	2.25	1.61	1.90
$f5$	11.46	1.98	2.38	1.70	1.98
$f6$	2.86	1.94	2.32	1.73	1.99
$f7$	11.75	3.33	3.63	3.00	3.27
$f8$	3.61	2.25	2.69	2.42	2.25
$f9$	4.57	2.04	2.61	1.86	2.09
$f10$	4.74	2.04	2.52	1.96	2.13
$f11$	3.68	2.23	2.52	2.00	2.12
$f12$	7.27	4.36	4.58	3.59	3.84
$f13$	8.73	4.20	4.55	3.62	3.80

3) Sensitivity to P_b: In the following, we study the effect of P_b, the percentage of best solutions used for the local surrogate model and the Powell method. In our experiments, P_b is set to be $0.02 \leq P_b \leq 0.5$. Fig. 10(b) presents the average number of FEs used to achieve the goal $f < 10^{-14}$. From the figure, it is clear that EDA/LS is not very sensitive to P_b on these test instances.

4) Sensitivity to P_c: To investigate the sensitivity of P_c, ten different values of P_c in [0.05, 0.5] are tested. Fig. 10(c) shows the statistical results. It is clear that EDA/LS with different P_c

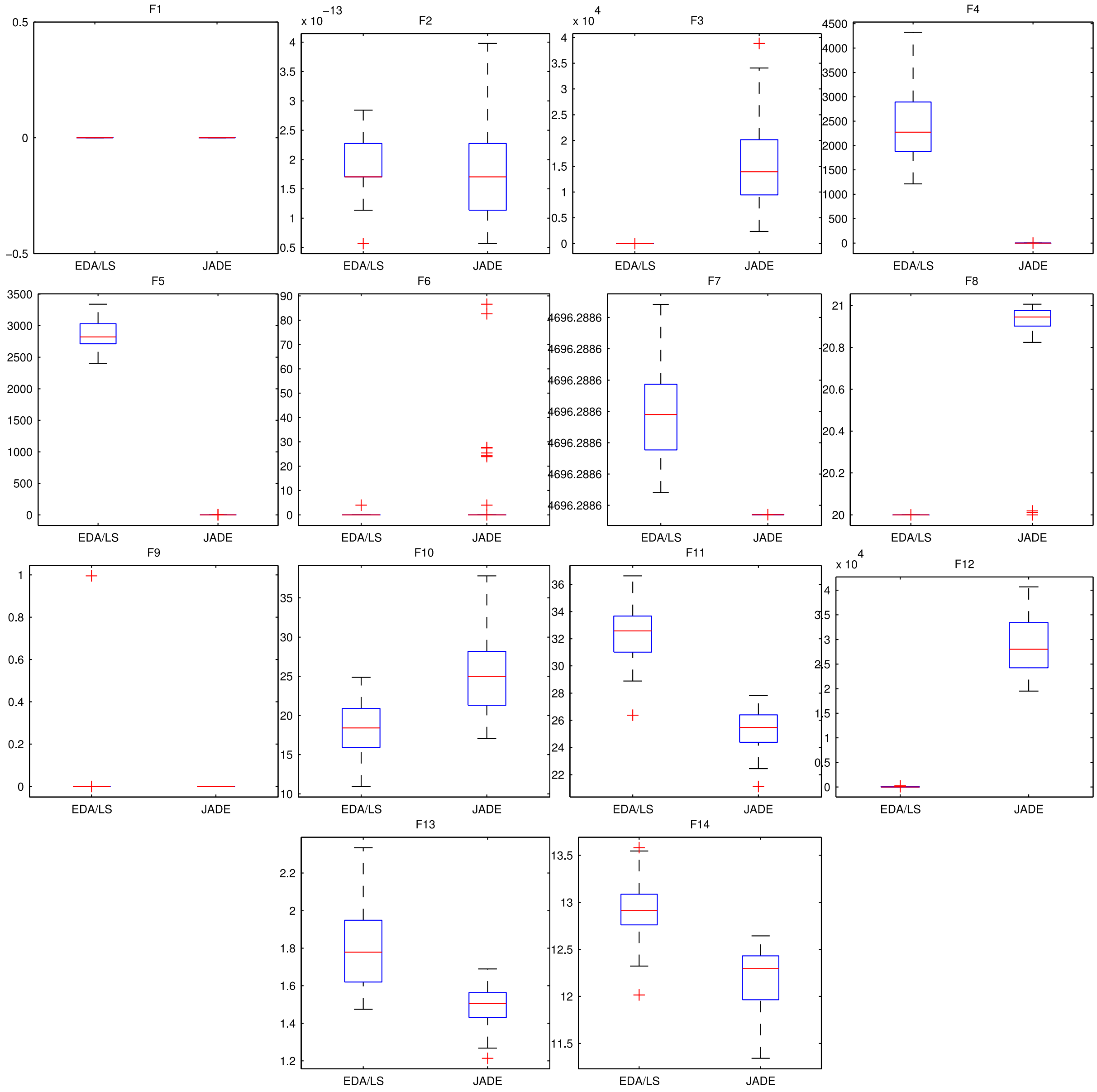


Fig. 11. Box plots of the final results obtained by EDA/LS and JADE on $F1$–$F14$.

has slightly different performances in terms of solution quality on $f2$ and $f5$, and although on $f3$ and $f4$, the performances of EDA/LS with different P_c values vary a little bit, the algorithm with different parameters can achieve the goal within 2.0×10^5 FEs.

5) Sensitivity to θ: To study this issue, ten different values of θ in the range $[0.01, 0.45]$ are tested. The results is shown in Fig. 10(d), which clearly shows that EDA/LS with different θ values performs quite similar.

G. Time Used in Search

The CPU time is another major concern when applying EAs. We record the average times used by the five algorithms in the experiments and present them in Table VI. It can be seen from this table that EDA/LS requires more CPU time than the other four algorithms. The time used by EDA/LS is 1.20 to 6.73 times of those used by other methods.

V. Performance on CEC05 Test Suite

The test suite used in the previous section can be handled well by both JADE and EDA/LS. In this section, we test the performance of EDA/LS on the first 14 instances in the CEC05 test suite [42], which is believed much harder. We denote them as $F1$–$F14$. $F1$–$F4$ are unimodal and $F5$–$F14$ are multimodal. The global optimum of all these test instances are 0.

In Section IV, we have shown that EDA/LS and JADE have the best performances among all the five methods in comparison. So we only compare EDA/LS and JADE on $F1$–$F14$. The variable number of all the test instances is 30 and all the algorithm parameters are the same as in Section IV-B.

The statistical results are shown in Table VII and Fig. 11. Fig. 11 shows that EDA/LS performs similar to JADE on $F1$, $F6$, and $F9$; EDA/LS is better on $F3$, $F8$, $F10$, and $F12$; while EDA/LS is worse on $F2$, $F4$, $F5$, $F7$, $F11$, $F13$, and $F14$. It should be noted that the optima of these instances are all 0.

TABLE VII
STATISTICAL RESULTS FOR EDA/LS AND JADE ON $F1$–$F14$. "+," "−," OR "∼" IN PARENTHESES INDICATES THE FUNCTION VALUE IS SMALLER THAN, GREATER THAN, OR SIMILAR TO THAT OBTAINED BY EDA/LS AT 95% SIGNIFICANCE LEVEL BY A WILCOXON RANK SUM TEST

	EDA/LS					JADE				
	1st Q	median	3rd Q	mean	std.	1st Q	median	3rd Q	mean	std.
$F1$	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00(∼)	0.00e+00
$F2$	1.71e-13	1.71e-13	2.27e-13	2.00e-13	4.63e-14	1.14e-13	1.71e-13	1.71e-13	**1.71e-13**(+)	6.60e-14
$F3$	1.71e-13	2.27e-13	2.27e-13	**1.03e-02**	7.31e-02	8.32e+03	1.39e+04	2.00e+04	1.57e+04(-)	8.79e+03
$F4$	1.83e+03	2.24e+03	2.61e+03	2.38e+03	7.49e+02	3.18e-12	6.71e-11	4.89e-10	**1.51e-09**(+)	5.52e-09
$F5$	2.71e+03	2.82e+03	3.00e+03	2.86e+03	2.32e+02	2.90e-05	5.95e-04	2.41e-03	**2.97e-02**(+)	1.48e-01
$F6$	1.14e-13	1.14e-13	1.71e-13	**7.97e-02**	5.64e-01	5.68e-14	1.14e-13	1.14e-13	6.12e+00(∼)	1.80e+01
$F7$	4.70e+03	4.70e+03	4.70e+03	4.70e+03	7.68e-10	4.70e+03	4.70e+03	4.70e+03	**4.70e+03**(+)	2.58e-12
$F8$	2.00e+01	2.00e+01	2.00e+01	**2.00e+01**	2.72e-07	2.09e+01	2.09e+01	2.10e+01	2.09e+01(-)	2.28e-01
$F9$	0.00e+00	0.00e+00	0.00e+00	1.99e-02	1.41e-01	0.00e+00	0.00e+00	0.00e+00	**0.00e+00**(∼)	0.00e+00
$F10$	1.59e+01	1.79e+01	2.09e+01	**1.83e+01**	3.25e+00	2.11e+01	2.47e+01	2.77e+01	2.47e+01(-)	4.78e+00
$F11$	3.09e+01	3.25e+01	3.37e+01	3.24e+01	2.02e+00	2.44e+01	2.55e+01	2.64e+01	**2.54e+01**(+)	1.45e+00
$F12$	1.71e-13	3.47e-07	1.15e+01	**1.77e+01**	4.54e+01	2.42e+04	2.79e+04	3.34e+04	2.90e+04(-)	5.25e+03
$F13$	1.61e+00	1.78e+00	1.93e+00	1.80e+00	2.08e-01	1.43e+00	1.50e+00	1.56e+00	**1.49e+00**(+)	1.03e-01
$F14$	1.27e+01	1.29e+01	1.31e+01	1.29e+01	3.24e-01	1.19e+01	1.23e+01	1.24e+01	**1.22e+01**(+)	3.28e-01

Table VII shows that in terms of the median values, EDA/LS fails on $F4$, $F5$, $F7$, $F8$, $F10$, $F11$, $F13$, and $F14$; while JADE fails on $F3$, $F5$, $F7$, $F8$, $F10$–$F14$.

A major difference between the CEC05 test suite and the YLL test suite is that the instances in the former are rotated problems with strong variable linkages. Table VII shows that EDA/LS fails on eight out of 14 instances. This could be because that in EDA/LS, both the VWH model and the cheap LS do not consider the variable linkages.

In summary, we may conclude that neither JADE nor EDA/LS can successfully solve all the instances, and the performances of JADE and EDA/LS are similar in terms of solution quality on this test suite.

VI. CONCLUSION

In this paper, we have proposed a new EA, named EDA/LS, for box-constrained global optimization. The basic ideas behind EDA/LS are that both the population distribution and individual location information should be used to guide the search, and other search techniques should be hybridized with simple models to improve the algorithmic performance. EDA/LS implements these two ideas by using a VWH model to extract and utilize global population distribution information, incorporating cheap LS to use solution location information for effective offspring reproduction, and applying expensive LS to improve promising solutions found so far when the EDA search has converged. In EDA/LS, the trial solutions are firstly sampled from the VWH model and then some components of the newly sampled solutions are replaced by those of the derived solutions from the local surrogate models.

EDA/LS has been compared with two state-of-the-art algorithms, JADE [49] and DE/EDA [38], and two pure histogram models based EDAs, called EDA/EWH and EDA/EHH on 13 widely used test instances from the YLL test suite [41]. The statistical results have shown that both EDA/EWH and EDA/EHH work poorly on most of the test instances, and DE/EDA can get good results on some test instances. Both EDA/LS and JADE can find high-quality solutions on all the test instances except $f7$, but EDA/LS is faster than JADE on most test instances.

The contributions of the major algorithm components have also been studied. The experimental results have suggested that the VWH model, the cheap LS, and the expensive LS are necessary in EDA/LS. The effect of the algorithm parameters of EDA/LS have been also empirically studied on four selected instances. The experimental results have indicated that EDA/LS can solve some instances with a large number of variables dimensions. Moreover, EDA/LS is not very sensitive to algorithm parameters.

Finally, we have compared EDA/LS and JADE on 14 hard instances from the CEC05 test suite [42]. Both of them have failed on more than half of the instances. But they are able to get high-quality solutions for the other instances.

There are several research avenues worthwhile exploring in the future. First, some other local surrogate models and classical LS methods should be investigated in our algorithm. Secondly, tuning the control parameters adaptively should be studied. Thirdly, some constraint handling strategies [36], [55] should be incorporated into EDA/LS for dealing with constrained optimization problems.

REFERENCES

[1] J. H. Holland, *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI, USA: Univ. Michigan Press, 1975.

[2] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Boston, MA, USA: Addison Wesley, 1989.

[3] K. De Jong, *Evolutionary Computation: A Unified Approach*. Cambridge, MA, USA: MIT Press, 2006.

[4] H. Mühlenbein and G. Paaß, “From recombination of genes to the estimation of distributions I: Binary parameters,” in *Parallel Problem Solving From Nature—PPSN IV* (LNCS 1141). Berlin, Germany: Springer, 1996, pp. 178–187.

[5] H. Mühlenbein, J. Bendisch, and H. M. Voigt, “From recombination of genes to the estimation of distributions II: Continuous parameters,” in *Parallel Problem Solving From Nature—PPSN IV* (LNCS 1141). Berlin, Germany: Springer, 1996, pp. 188–197.

[6] J. A. Lozano, P. Larrañga, I. Inza, and E. Bengoetxea, Eds., *Towards a New Evolutionary Computation: Advances in Estimation of Distribution Algorithms*, vol. 192. Berlin, Germany: Springer, 2006.

[7] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York, NY, USA: Springer, 2006.

[8] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. New York, NY, USA: Springer, 2001.

[9] M. Pelikan, D. E. Goldberg, and F. G. Lobo, “A survey of optimization by building and using probabilistic models,” *Comput. Optim. Appl.*, vol. 21, no. 1, pp. 5–20, 2002.

[10] P. Larrañaga and J. A. Lozano, Eds., *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Boston, MA, USA: Kluwer Academic, 2002.

[11] M. Pelikan, K. Sastry, and E. Cantu-Paz, Eds., *Scalable Optimization via Probabilistic Modeling: From Algorithms to Applications*. Berlin, Germany: Springer, 2006.

[12] S. Zhou and Z. Sun, “A survey on estimation of distribution algorithms,” *Acta Autom. Sin.*, vol. 32, no. 2, pp. 113–124, 2007.

[13] S. Baluja, “Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning,” Dept. Comput. Sci., Carnegie Mellon Univ., Pittsburgh, PA, USA, Tech. Rep. CUM-CS-94-163, 1994.

[14] P. A. N. Bosman and D. Thierens, “Expanding from discrete to continuous estimation of distribution algorithms: The IDEA,” in *Parallel Problem Solving from Nature—PPSN VI*. Berlin, Germany: Springer, 2000, pp. 767–776.

[15] G. R. Harik, F. G. Lobo, and D. E. Goldberg, “The compact genetic algorithm,” *IEEE Trans. Evol. Comput.*, vol. 3, no. 4, pp. 287–297, Nov. 1999.

[16] P. Larrañaga, R. Etxeberria, J. A. Lozano, and J. M. Peña, “Optimization in continuous domains by learning and simulation of Gaussian networks,” in *Proc. Genet. Evol. Comput. Conf. (GECCO 2010)*, Las Vegas, NV, USA, pp. 201–204.

[17] J. S. De Bonet, C. L. Isbell, and P. Viola, “MIMIC: Finding optima by estimating probability densities,” in *Advances in Neural Information Processing Systems*. Cambridge, MA, USA: MIT Press, 1997, pp. 424–431.

[18] S. Baluja and S. Davies, “Using optimal dependency-trees for combinatorial optimization: Learning the structure of the search space,” in *Proc. 14th Int. Conf. Mach. Learn.*, Nashville, TN, USA, 1997, pp. 30–38.

[19] H. Mühlenbein and T. Mahnig, “FDA—A scalable evolutionary algorithm for the optimization of additively decomposed functions,” *Evol. Comput.*, vol. 7, no. 4, pp. 353–376, 1999.

[20] M. Pelikan, D. Goldberg, and E. Cantú-Paz, “BOA: The Bayesian optimization algorithm,” Dept. Gen. Eng., Illinois Genetic Algorithms Laboratory (IlliGAL), Urbana, IL, USA, Tech. Rep. 99003, 1999.

[21] R. Etxeberria and P. Larrañaga, “Global optimization using Bayesian networks,” in *Proc. 2nd Symp. Artif. Intell. (CIMAF-99)*, Habana, Cuba, pp. 332–339.

[22] N. Hansen and A. Ostermeier, “Completely derandomized self-adaptation in evolution strategies,” *Evol. Comput.*, vol. 9, no. 2, pp. 159–195, 2001.

[23] Q. Zhang, J. Sun, E. Tsang, and J. Ford, “Hybrid estimation of distribution algorithm for global optimization,” *Eng. Comput.*, vol. 21, no. 1, pp. 91–107, 2004.

[24] W. Dong, T. Chen, P. Tino, and X. Yao, “Scaling up estimation of distribution algorithms for continuous optimization,” *IEEE Trans. Evol. Comput.*, vol. 17, no. 6, pp. 797–822, Dec. 2013.

[25] H. Wu and J. L. Shapiro, “Does overfitting affect performance in estimation of distribution algorithms,” in *Proc. Genet. Evol. Comput. Conf. (GECCO)*, vol. 2. Seattle, WA, USA: ACM, 2006, pp. 433–434.

[26] M. Pelikan and H. Mühlenbein, “The bivariate marginal distribution algorithm,” in *Advances in Soft Computing—Engineering Design and Manufacturing*. London, U.K.: Springer, 1999, pp. 521–535.

[27] Q. Lu and X. Yao, “Clustering and learning Gaussian distribution for continuous optimization,” *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 35, no. 2, pp. 195–204, May 2005.

[28] B. T. Zhang, J. Yang, and S. W. Chi, “Self-organizing latent lattice models for temporal gene expression profiling,” *Mach. Learn.*, vol. 52, nos. 1–2, pp. 67–89, 2003.

[29] D. Y. Cho and B. T. Zhang, “Continuous estimation of distribution algorithms with probabilistic principal component analysis,” in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Seoul, Korea: IEEE Press, 2001, pp. 521–526.

[30] D. Y. Cho and B. T. Zhang, “Evolutionary optimization by distribution estimation with mixtures of factor analyzers,” in *Proc. IEEE Congr. Evol. Comput. (CEC)*. Honolulu, HI, USA: IEEE Press, 2002, pp. 1396–1401.

[31] D. Y. Cho and B. T. Zhang, “Evolutionary continuous optimization by distribution estimation with variational Bayesian independent component analyzers mixture model,” in *Parallel Problem Solving from Nature—PPSN VIII* (LNCS 3242). Birmingham, U.K.: Springer, 2004, pp. 212–221.

[32] Q. Zhang, J. Sun, and E. Tsang, “An evolutionary algorithm with guided mutation for the maximum clique problem,” *IEEE Trans. Evol. Comput.*, vol. 9, no. 2, pp. 192–200, Apr. 2005.

[33] A. Zhou, Q. Zhang, Y. Jin, and B. Sendhoff, “Combination of EDA and DE for continuous biobjective optimization,” in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Hong Kong: IEEE Press, 2008, pp. 1447–1454.

[34] J. A. Vrugt and B. A. Robinson, “Improved evolutionary optimization from genetically adaptive multimethod search,” in *Proc. Nat. Acad. Sci. (PNAS)*, vol. 104, no. 3, pp. 708–711, 2007.

[35] Q. Zhang, J. Sun, and E. Tsang, “Combinations of estimation of distribution algorithms and other techniques,” *Int. J. Autom. Comput.*, vol. 4, no. 3, pp. 273–280, 2007.

[36] J. Sun and J. Garibaldi, “A novel memetic algorithm for constrained optimization,” in *Proc. World Congr. Comput. Intell.*, Barcelona, Spain, 2010.

[37] J. Sun, J. Garibaldi, N. Krasnogor, and Q. Zhang, “An intelligent multi-restart memetic algorithm for box constrained global optimization,” *J. Evol. Comput.*, vol. 21, no. 1, pp. 107–147, 2013.

[38] J. Sun, Q. Zhang, and E. Tsang, “DE/EDA: A new evolutionary algorithm for global optimization,” *Inf. Sci.*, vol. 169, no. 3, pp. 249–262, 2005.

[39] R. Fletcher, *Practical Methods of Optimization*. Chichester, U.K.: Wiley, 1987.

[40] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes: The Art of Scientific Computing*. Cambridge, U.K.: Cambridge Univ. Press, 2007.

[41] X. Yao, Y. Liu, and G. Lin, “Evolutionary programming made faster,” *IEEE Trans. Evol. Comput.*, vol. 3, no. 2, pp. 82–102, Jul. 1999.

[42] P. N. Suganthan *et al.*, “Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization,” Dept. Comput. Sci., Nanyang Technol. Univ., Singapore; Indian Inst. Technol., Kanpur, India, Tech. Rep. 2005005, 2005.

[43] M. Pelikan, D. E. Goldberg, and S. Tsutsui, “Getting the best of both worlds: Discrete and continuous genetic and evolutionary algorithms in concert,” *Inf. Sci.*, vol. 156, nos. 3–4, pp. 147–171, 2003.

[44] P. Posik, “On the use of probabilistic models and coordinate transforms in real-valued evolutionary algorithms,” Ph.D. dissertation, Dept. Cybern., Czech Tech. Univ., Prague, Czech Republic, 2007.

[45] B. Liu, Q. Zhang, and G. G. E. Gielen, “A Gaussian process surrogate model assisted evolutionary algorithm for medium scale expensive optimization problems,” *IEEE Trans. Evol. Comput.*, vol. 18, no. 2, pp. 180–192, Apr. 2014.

[46] M. J. D. Powell, “UOBYQA: Unconstrained optimization by quadratic approximation,” *Math. Program.*, vol. 92, no. 3, pp. 555–582, 2002.

[47] Y. Jiao, C. Dang, Y. Leung, and Y. Hao, “A modification to the new version of the price’s algorithm for continuous global optimization problems,” *J. Glob. Optim.*, vol. 36, pp. 609–626, 2006.

[48] E. F. Wanner, A. Frederico, G. Guimar, R. H. C. Takahashi, and P. J. Fleming, “Local search with quadratic approximations into memetic algorithms for optimization with multiple criteria,” *Evol. Comput.*, vol. 16, no. 2, pp. 185–224, 2008.

[49] J. Zhang and A. C. Sanderson, “JADE: Adaptive differential evolution with optional external archive,” *IEEE Trans. Evol. Comput.*, vol. 13, no. 5, pp. 945–958, Oct. 2009.

[50] R. Storn and K. Price, “Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces,” *J. Glob. Optim.*, vol. 11, no. 4, pp. 341–359, 1997.

[51] J. Brest, S. Greiner, B. Boskovic, M. Mernik, and V. Zumer, "Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems," *IEEE Trans. Evol. Comput.*, vol. 10, no. 2, pp. 646–657, Dec. 2006.

[52] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *IEEE Trans. Evol. Comput.*, vol. 13, no. 2, pp. 398–417, Apr. 2009.

[53] I. C. Trelea, "The particle swarm optimization algorithm: Convergence analysis and parameter selection," *Inf. Process. Lett.*, vol. 85, no. 6, pp. 317–325, 2003.

[54] Y. Jin and J. Branke, "Evolutionary optimization in uncertain environments—A survey," *IEEE Trans. Evol. Comput.*, vol. 9, no. 3, pp. 303–317, Jun. 2005.

[55] Y. Wang, Z. Cai, Y. Zhou, and W. Zeng, "An adaptive tradeoff model for constrained evolutionary optimization," *IEEE Trans. Evol. Comput.*, vol. 12, no. 1, pp. 80–92, Feb. 2008.

Aimin Zhou (S'08–M'10) received the B.Sc. and M.Sc. degrees in computer science from Wuhan University, Wuhan, China, in 2001 and 2003, respectively, and the Ph.D. degree in computer science from University of Essex, Colchester, U.K., in 2009.

He is an Associate Professor with the Department of Computer Science and Technology, East China Normal University, Shanghai, China. His research interests include evolutionary computation, machine learning, image processing, and their applications. He has published over 30 peer-reviewed papers on evolutionary computation and image processing.

Jianyong Sun received the B.Sc. degree in computational mathematics from Xi'an Jiaotong University, Shaanxi, China, in 1997, and the M.Sc. and Ph.D. degrees in applied mathematics and computer science from University of Essex, Colchester, U.K., in 1999 and 2005, respectively.

He is a Senior Lecturer with the Faculty of Engineering and Science, University of Greenwich, Greenwich, U.K. His research interests include evolutionary computation and statistical machine learning, and their applications in computational biology, bioinformatics, image processing, and big data analytics. He has published over 30 peer-reviewed papers on evolutionary algorithms and statistical machine learning in prestigious journals such as IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, IEEE TRANSACTIONS ON COMPUTATIONAL BIOLOGY AND BIOINFORMATICS, and *PNAS*.

Qingfu Zhang (M'01–SM'06) received the B.Sc. degree in mathematics from Shanxi University, Taiyuan, China, in 1984, and the M.Sc. and Ph.D. degrees in applied mathematics and information engineering from Xidian University, Xi'an, China, in 1991 and 1994, respectively.

He is a Professor with the Department of Computer Science, City University of Hong Kong, Hong Kong, and the School of Computer Science and Electronic Engineering, University of Essex, Colchester, U.K. He is also a Changjiang Visiting Chair Professor with Xidian University. He is currently leading the Metaheuristic Optimization Research Group, City University of Hong Kong. His research interests include evolutionary computation, optimization, neural networks, data analysis, and their applications. He has authored several research publications and holds two patents.

Dr. Zhang won the Unconstrained Multiobjective Optimization Algorithm Competition at the Congress of Evolutionary Computation 2009 for a multiobjective optimization algorithm developed in his group and the 2010 IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION Outstanding Paper Award. He is an Associate Editor of IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION and IEEE TRANSACTIONS ON CYBERNETICS. He is also an Editorial Board Member of three international journals.