Hybrid Estimation of Distribution Algorithm for Global Optimization

Qingfu Zhang, Jianyong Sun, Edward Tsang and John Ford [*]
Department of Computer Science, University of Essex,
Wivenhoe Park, Colchester, CO4 3SQ, U K
E-mail: qzhang@essex.ac.uk

Keywords Evolutionary Algorithms, Estimation of Distribution Algorithm, Global Optimization, Simplex Method, Quadratic Approximation.

Abstract This paper introduces a new hybrid evolutionary algorithm for continuous global optimization problems, called Estimation of Distribution algorithm with Local search (EDA/L). Like other evolutionary algorithms, EDA/L maintains and improves a population of solutions in the feasible region. Initial candidate solutions are generated by uniform design, these solutions evenly scatter over the feasible solution region. To generate a new population, a marginal histogram model is built based on the global statistical information extracted from the current population, and then new solutions are sampled from the model thus built. The incomplete simplex method applies to every new solution generated by uniform design or sampled from the histogram model. UOBDQA (unconstrained optimization by diagonal quadratic approximation) applies to several selected resultant solutions of the incomplete simplex method at each generation. We study the effectiveness of main components of EDA/L. The experimental results demonstrate that EDA/L is better than four other recent evolutionary algorithms in terms of the solution quality and the computational cost.

1 Introduction

In the paper we are considering the following global optimization problem:

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & x \in D \end{array} \tag{1}$$

where $x = (x_1, x_2, \cdots, x_n) \in R^n$, $f : R^n \to R$ is the objective function and $D = \{x \mid a_i \le x_i \le b_i \text{ for } i = 1, 2, \cdots, n\}$ is the feasible region. This

[*]This work was supported by EPSRC under Grant GR/R64742/01 and a Research Promotion Fund grant from the University of Essex.

problem arises in almost every field of science, engineering and business. Often, the objective function $f(x)$ is non-convex and has many local minima in the feasible region that are not the global minimum. A traditional optimization algorithm [8] such as the conjugate gradient method and the downhill simplex method could fail in locating the global minimum since such an algorithm has no mechanism to escape from a local minimum once it has been trapped in it. The global optimization problem becomes even more difficult if derivatives of $f(x)$ are unavailable as in many applications. Unless special assumptions are made about the objective function f, any algorithm, in general, has to evaluate f on a dense subset of the feasible region to find the global minimum [17]. This intractability makes room for developing heuristic algorithms such as evolutionary algorithms (EA) for the global optimization problem. Indeed, many evolutionary algorithms have been developed for this problem in the past [1][6][10][19].

Estimation of distribution algorithms (EDAs) [4][5][9][14][20] are a new class of evolutionary algorithms. Like other evolutionary algorithms, EDAs maintain and successively improve a population of potential solutions until some stopping condition is met. However, EDAs do not use crossover or mutation. Instead, they select the best solutions from the current population and explicitly extract global statistical information from the selected solutions. A posterior probability distribution model of promising solutions is built, based on the extracted information. Then new solutions are sampled from the model thus built and fully or in part replace solutions in the current population. More precisely, EDAs work as follows:

Step 0: Randomly pick a set of solutions to form the initial population.

Step 1: Select some solutions from the current population according to a selection method. Build the probability model of the selected solutions.

Step 2: Replace some or all of the members of the current population by new solutions sampled from the probability model.

Step 3: If the stopping condition are not met, go to **Step 1**.

Several EDAs have been proposed for solving global optimization problems. In these existing algorithms, the probability distribution of the promising solutions are modelled by a Gaussian distribution [9], a Gaussian mixture [4] or a histogram [18]. Since many points are needed to build a good probability model, these algorithms are often very time-consuming in practice.

A very natural way to improve the performance of EAs is to hybridize local search with EAs. Indeed, the hybrid evolutionary approach, often called the memetic algorithm in the literature, has proved to be a very efficient algorithmic framework for solving hard optimization problems, particularly, hard combinatorial optimization problems (e.g., [2][11][16]). In a hybrid evolutionary algorithm, EA operators (such as crossover and mutation) generate a starting point, a local search method starts from this point and locates the corresponding local minimum. Due to the diversity provided by EA operators, the algorithm is less likely to be trapped in a local minimum, while the local search speeds up the convergence of the algorithm greatly. Very recently, Bosman and Thierens have hybridized their iterated density estimation evolutionary algorithm (IEDA) with the conjugate gradient algorithm for solving the global optimization problem [3]. Their algorithm requires calculation of derivatives so it can not be used in the case when the derivatives of the objective function $f(x)$ are not available.

In this paper, we introduce a new hybrid EDA for global optimization which has been successful on a number of test problems. We refer to the algorithm as Estimation of Distribution Algorithm with Local search (EDA/L). Most existing hybrid evolutionary algorithms utilize one local search technique. In EDA/L approach, however, we use two local search algorithms. One is the incomplete simplex method [13] and the other is UOBDQA (Unconstrained Optimization By Diagonal Quadratic Approximation) [15]. For every new solution that is generated in the initialization step or sampled from a probability model in EDA/L, we apply the downhill simplex method to it and only allow the simplex method to run $O(n)$ steps. Then if the resultant solution of the incomplete simplex method is good enough, we apply UOBDQA to it. The incomplete simplex method helps to locate promising solutions more accurately while the role of UOBDQA is to find good local minima and hopefully the global minimum. The other feature of EDA/L is that it generates its initial population from D by the uniform design technique [7]. The uniform design technique can generate points in D that are more evenly distributed in D than points generated randomly. We study the effectiveness of main components of our proposed algorithm in this paper. The experimental results show that our algorithm is better than four recent evolutionary algorithms in terms of the solution quality and the computational cost.

2 The Algorithm

2.1 The Framework of the Algorithm

The framework of the proposed algorithm is as follows:

Step 0 Parameter Setting Population size: N, the best solution value found so far: $Fbest = \infty$, the number of new solutions sampled from the probability model at each iteration: K, the maximal number of function evaluations in the simplex method: S, and the number of solutions undergoing UOBDQA at each iteration: J. $J < K$.

Step 1 Initialization Generate N solutions $\widetilde{x}^1, \widetilde{x}^2, \cdots, \widetilde{x}^N$ from D, using the uniform design technique. Apply the simplex method with at most S function evaluations to each solution $\widetilde{x}^i$ and obtain $x^i (1 \leq i \leq N)$. Then let $x^1, x^2, \cdots, x^N$ constitute the initial population.

Step 2 Reproduction Build a probability model based on the statistical information extracted from some selected solutions in the current population. Sample K new solutions $\widetilde{x}^{N+1}, \cdots \widetilde{x}^{N+K}$ from this model and then apply the simplex method with at most S function evaluations to each solution $\widetilde{x}^i$ and obtain x^i $(N + 1 \leq i \leq N + K)$.

Step 3 Comparison Compare the function values of all x^i $(1 \leq i \leq N + K)$, order and relabel them such that

$$f(x^1) \leq f(x^2) \leq \cdots \leq f(x^{N+K}).$$

Step 4 Update of *Fbest* Apply UOBDQA to x^i $(1 \leq i \leq J)$ and obtain J local minima y^i $(1 \leq i \leq J)$. If $Fbest > \min_{1 \leq i \leq J} f(y^i)$, set $Fbest = \min_{1 \leq i \leq J} f(y^i)$.

Step 5 Stopping Condition If the stopping condition is met, stop.

Step 6 Update of the Population Let $x^{J+1}, \cdots, x^{J+N}$ constitute new population. Go to Step 2.

Experimental design techniques [12] such as orthogonal design and uniform design have been proposed to improve the performance of genetic algorithms [10][21]. The total number of orthogonal design points is often much more than 2^n while the number of uniform design points is relatively smaller. This is the reason why we use the uniform design in Initialization (Step 1). The probability model built in Reproduction (Step 2) step models the distribution of the best solutions in the current population. Therefore, sampling solutions from this model should fall in promising areas with high probability. In Step 3 (Comparison), since all the solutions have undergone the incomplete simplex method with at most S function value evaluations, the best ones, i.e., $x^1, x^2, \cdots, x^J$ in Step 4, should be more likely to be close to the global optimum than other solutions. We apply UOBQDA only to these best solutions in Step 4. The details of the main ingredients of EDA/L are explained in the following.

2.2 Initialization

Before we solve a global optimization, we often have no information about the location of the global minimum of the objective function. Therefore, the solutions $\widetilde{x}^1, \widetilde{x}^2, \cdots, \widetilde{x}^N$ in Step 1 should scatter over the feasible region D as uniformly as possible. The most popular measure of uniformity is L_p-discrepancy. Let $\mathcal{P} = \{z^1, z^2, \cdots, z^N\}$ be a set of N points in $C = [0,1]^n$, The L_p-discrepancy of P over C is defined as follows [7]:

$$D_p(\mathcal{P}) = \left\{ \int_C \left| \frac{\mathcal{N}(\mathcal{P}, [0,x))}{n} - Vol([0,x)) \right|^p dx \right\}^{1/p}$$

where $[0,x) = [0,x_1) \times [0,x_2) \times \cdots \times [0,x_n)$, $\mathcal{N}(\mathcal{P}, [0,x))$ is the number of points of $\mathcal{P}$ falling in $[0,x)$ and $Vol([0,x) = \prod_{i=1}^n x_i$ is the volume of $[0,x)$. The goal of uniform design is to generate $z^1, z^2, \cdots, z^N$ to minimize $D_p(\mathcal{P})$. Uniform design provides a series of uniform arrays for different n and q. A uniform array is often denoted by $U_N(q^n)$. The following is an example of uniform arrays:

$$U_9(9^6) = \begin{bmatrix} 1 & 2 & 4 & 5 & 7 & 8 \\ 2 & 4 & 8 & 1 & 5 & 7 \\ 3 & 6 & 3 & 6 & 3 & 6 \\ 4 & 8 & 7 & 2 & 1 & 5 \\ 5 & 1 & 2 & 7 & 8 & 4 \\ 6 & 3 & 6 & 3 & 6 & 3 \\ 7 & 5 & 1 & 8 & 4 & 2 \\ 8 & 7 & 5 & 4 & 2 & 1 \\ 9 & 9 & 9 & 9 & 9 & 9 \end{bmatrix}$$

There are N rows in $U_N(q^n)$ and each row represents a point in R^n. All these points are integer points in $[1,q]^n$. Let $u^k = (u_1^k, \cdots, u_n^k)$ $(1 \leq k \leq N)$ be points (rows) in $U_N(q^n)$, we can generate $z^1, z^2, \cdots, z^N$ in $[0,1]^n$ in the following way:

$$z^i = (\frac{2u_1^i - 1}{2q}, \frac{2u_2^i - 1}{2q}, \cdots, \frac{2u_n^i - 1}{2q}), \qquad 1 \leq i \leq N.$$

The discrepancy of these points generated in this way is much smaller than that of s randomly generated points over C. In the Initialization step, based on $U_N(q^n)$, N points $\widetilde{x}^1, \widetilde{x}^2, \cdots, \widetilde{x}^N$ can be generated as follows:

$$z^i = (a_1 + \frac{2u_1^i - 1}{2q}(b_1 - a_1), a_2 + \frac{2u_2^i - 1}{2q}(b_2 - a_2), \cdots, a_n + \frac{2u_n^i - 1}{2q}(b_n - a_n)),$$
$$1 \leq i \leq N.$$

Many uniform design arrays can be found from http:// www.math.hkbu.edu.hk /UniformDesign. There are several algorithms for generating uniform design arrays [7].

2.3 Reproduction

The role of reproduction operators such as crossover and mutation in evolutionary algorithms is to generate new promising solutions in the search space.

The reproduction operator in EDAs generates new solutions by sampling from a probability model which characterizes the distribution of promising solutions. Several different probability models have been proposed in EDAs for continuous optimization problems. A Gaussian model can be built with very low computational cost but cannot be used to cope with the multimodal objective functions [9]. A Gaussian mixture model can deal with any multimodal objective function in theory but the cost of building such a model is prohibitively high, particularly for large scale problems [3]. We employ the marginal histogram model [18] in EDA/L. The overhead of building this model is $O(n)$. Therefore, it is suitable for solving large scale problems. Since the variables are independent of one another in the marginal histogram model, its modelling ability is poorer than that of a Gaussian mixture model. However, the utilization of local search in EDA/L can offset this shortcoming.

2.3.1 Histogram model

Let $x^1, x^2, \cdots, x^N$ be the current population, we select the M best solutions from the current population. Without loss of generality, we assume that the M selected solutions are $x^1, x^2, \cdots, x^M$. To model the distribution of these M best solutions by a fixed-width histogram distribution [18], we divide the search space $[a_i, b_i]$ of each variable x_i into H subintervals with the same length. The $j-$th subinterval is[1]

$$[a_i + \frac{j-1}{H}(b_i - a_i), a_i + \frac{j}{H}(b_i - a_i)), (1 \leq j \leq H).$$

Then we count $W(i,j)$, the number of the selected solutions whose values of x_i fall in the $j-$th subinterval. The marginal probability density of x_i is modelled by

$$p_i(x_i) = \begin{cases} \frac{W(i,1)}{M}\frac{H}{b_i-a_i} & a_i \leq x_i < a_i + \frac{1}{H}(b_i - a_i) \\ \frac{W(i,2)}{M}\frac{H}{b_i-a_i} & a_i + \frac{1}{H}(b_i - a_i) \leq x_i < a_i + \frac{2}{H}(b_i - a_i) \\ \vdots & \vdots \\ \frac{W(i,H)}{M}\frac{H}{b_i-a_i} & a_i + \frac{H-1}{H}(b_i - a_i) \leq x_i \leq b_i \end{cases} \quad (2)$$

Figure 1 shows an example of marginal distribution.

2.3.2 Sampling method

A new solution $\widetilde{x} = (\widetilde{x}_1, \widetilde{x}_2, \cdots, \widetilde{x}_n)$ is generated by sampling the value of each x_i from (2) independently. To generate $\widetilde{x}_i$, we first randomly select a subinterval $[a_i + \frac{j-1}{H}(b_i - a_i), a_i + \frac{j}{H}(b_i - a_i))$ with probability $\frac{W(i,j)}{M}$, and then pick up a number from this subinterval with uniform distribution. In Reproduction, we generate K new solutions in this way.

[1]Precisely, the $H-$th subinterval is $[a_i + \frac{H-1}{H}(b_i - a_i), b_i]$.

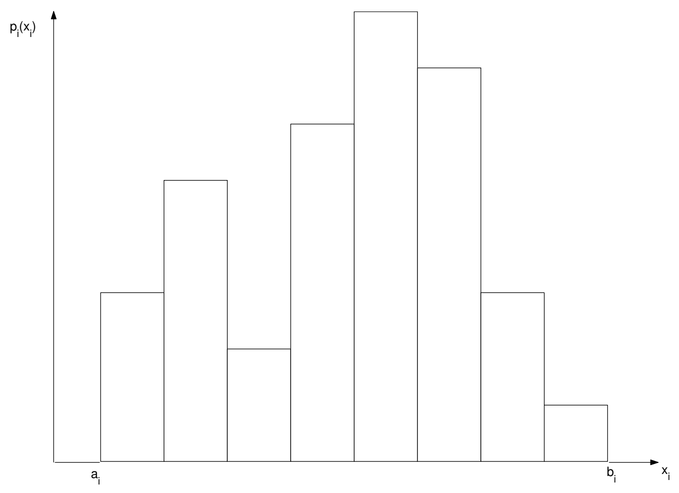

Figure 1: An example of marginal distribution.

2.4 Simplex Method

A simplex in R^n is the convex polytope with $n + 1$ affinely independent vertices. A simplex in R^2 is a triangle, in R^3 a tetrahedron, and so on. The downhill simplex method [8][13], introduced by Nelder and Mead in 1965, maintains at each iteration a simplex in R^n. Starting with a point v, this method generates other n points to form the initial simplex. Let $v^1, v^2, \cdots, v^{n+1}$ be the vertices of the current simplex, the simplex method will first order and relabel these vertices such that

$$f(v^1) \leq f(v^2) \leq \cdots \leq f(v^{n+1}),$$

v^1 is referred to as the best point, v^{n+1} the worst point and v^n the second worst point. Then a single new point, often called the accepted point, will be produced to replace the current worst vertex v^{n+1} in the set of the vertices, or n points will be generated, together with the current best point v^1 , to form the simplex at the next iteration. One iteration of the simplex method is given as follows:

Step 1 Ordering order the $n+1$ vertices such that $f(v^1) \leq f(v^2) \leq \cdots \leq f(v^{n+1})$.

Step 2 Reflection Compute the reflection point v^{ref}

$$v^{ref} = \overline{v} + (\overline{v} - v^{n+1}),$$

where $\overline{v} = \frac{1}{n} \sum_{i=1}^{n} v^i$ is the centroid of the n best vertices. If $f(v^1) \leq f(v^{ref}) < f(v^n)$, replace v^{n+1} by v^{ref} and terminate the iteration.

Step 3 Expansion If v^{ref} is better than v^1, i.e., $f(v^{ref}) < f(v^1)$, compute the expansion point v^{ex} :

$$v^{ex} = \overline{v} + 2(v^{ref} - \overline{v}),$$

replace v^{n+1} by the better of v^{ex} and v^{ref} and terminate the iteration.

Step 4 Inside Contraction If v^{ref} is not better than v^{n+1}, i.e., $f(v^{ref}) \geq f(v^{n+1})$, compute

$$v^{ic} = \overline{v} - \frac{1}{2}(\overline{v} - v^{n+1}),$$

if $f(v^{ic}) < f(v^{n+1})$, replace v^{n+1} by v^{ic} and terminate the iteration; otherwise, go to Step 6.

Step 5 Outside Contraction If v^{ref} is better than v^{n+1}, i.e., $f(v^n) \leq f(v^{ref}) < f(v^{n+1})$, compute

$$v^{oc} = \overline{v} + \frac{1}{2}(v^{ref} - \overline{v}),$$

if $f(v^{oc}) \leq f(v^{ref})$, replace v^{n+1} by v^{oc} and terminate the iteration; otherwise go to Step 6.

Step 6 Shrinking Compute

$$u^i = v^1 + \frac{1}{2}(v^i - v^1), i = 2, 3, \cdots, n+1.$$

Replace $v^2, v^3, \cdots, v^{n+1}$ by $u^2, u^3, \cdots, u^{n+1}$ and terminate the iteration.

2.5 UOBDQA

Trust region methods using quadratic interpolation model, developed by Powell [15], is a new class of derivative free local optimization algorithms for finding a local minimum of the objective function $f(x)$. Such algorithms start with an initial point v, ρ_{beg} and ρ_{end}, the initial and final values of a trust region radius ρ. The algorithms generate a set of interpolation points (including v) in a neighborhood of v. Then an initial quadratic model is formed by interpolating these points. The algorithm generate a new point, either by minimizing the current quadratic model within a trust region, or by a procedure that improves the accuracy of the model. One of the interpolation points is replaced by the resultant point. The typical distance between successive points at which f is calculated are of magnitude of the trust region radius ρ. The initial value of ρ is set to be ρ_{beg}. ρ will be reduced when the objective function stops decreasing for such changes to the points. The algorithms stop when ρ is smaller than $\rho_{end.}$

Several instances of the trust region methods have been implemented by Powell [15]. UOBYQA (Unconstrained Optimization By Quadratic Approximation), as its name suggests, builds a general quadratic model in a trust region. It needs to maintain $\frac{(n+1)(n+2)}{2}$ interpolation points at each iteration. The amount of routine work of an iteration takes $O(n^4)$. Consequently, UOBYQA will be prohibitive for the large value of n. To overcome this shortcoming, UOBDQA (Unconstrained Optimization By Diagonal Quadratic Approximation) was proposed, which chooses a quadratic of the form

$$Q(v^* + d) = f(v^*) + g^T d + d^T Dd$$

where v^* is the point with the least objective function among the current interpolation points. g is a vector and D is a diagonal matrix of dimension n. Therefore, $2n+1$ interpolation points are needed to determine this quadratic model. The amount of routine work of each iteration is $O(n^2)$ instead of $O(n^4)$ in UOBYQA, which makes UOBDQA more useful than UOBYQA for solving large scale optimization problems. For the detailed description of UOBYQA and UOBDQA, please refer to [15].

3 Numerical Experiments and Results

3.1 Test Suite

The following well-known functions are used in our experimental studies. All these functions are non-convex functions with many local minima.

$$f_1 = \sum_{i=1}^{n} (-x_1 \sin(\sqrt{|x_i|})$$

$$f_2 = \sum_{i=1}^{n} (-x_i^2 - 10\cos(2\pi x_i) + 10)$$

$$f_3 = -20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}) - \exp(\sum_{i=1}^{n} \cos(2\pi x_i))$$

$$f_4 = \frac{1}{4000}\sum_{i=1}^{n} (-x_i^2 - \prod_{i=1}^{n} \cos(\frac{x_i}{\sqrt{i}}) + 1)$$

$$f_5 = \frac{\pi}{n}\{10\sin^2(\pi y_i) + \sum_{i=1}^{n-1} (y_i - 1)^2[1 + 10\sin^2(\pi y_{i+1})] + (y_n - 1)^2\}$$

$$+ \sum_{i=1}^{n} u(x_i, 10, 100, 4)$$

where

$$y_i = 1 + \frac{1}{4}(x_i + 1)$$

and

$$u(x_i,a,k,m)=\begin{cases} k(x_i-a)^m & x_i>a \\ 0 & -a\le x_i\le a \\ k(-x_i-a)^m & x_i<-a \end{cases}$$

$f_6=\frac{1}{10}\{\sin^2(3\pi x_1)+\sum\limits_{n=1}^{n-1}(x_i-1)^2[1+\sin^2(3\pi x_{i+1})]$
$\quad +(x_n-1)^2(1+\sin^2(2\pi x_n)]\}+\sum\limits_{i=1}^{n}u(x_i,5,100,4)$

$f_7=-\sum\limits_{i=1}^{n}\sin(x_i)\sin^m(\frac{i\times x_i^2}{\pi})$

$f_8=-\sum\limits_{i=1}^{n}\left[\sum\limits_{j=1}^{n}(a_{ij}\sin\omega_j+b_{ij}\cos\omega_j)-\sum\limits_{j=1}^{n}(a_{ij}\sin x_j+b_{ij}\cos\omega_j)\right]$

where a_{ij} and b_{ij} are random integers in [-100,100], and ω_j is a random number in [-π,π].

$f_9=\frac{1}{n}\sum\limits_{i=1}^{n}(x_i^4-16x_i^2+5x_i)$

$f_{10}=\sum\limits_{i=1}^{n-1}[100(x_i^2-x_{i+1})^2+(x_j-1)^2).$

The basic characteristics of these test functions are given in Table 1.

Table 1: The basic characteristics of the test functions

Test Function	Globally minimal function value	Feasible Solution Region	n
f_1	-12569.5	$[-500,500]^n$	30
f_2	0	$[-5.12,5.12]^n$	30
f_3	0	$[-32,32]^n$	30
f_4	0	$[-600,600]^n$	30
f_5	0	$[-50,50]^n$	30
f_6	0	$[-50,50]^n$	30
f_7	-99.2784	$[0,\pi]^n$	100
f_8	0	$[-\pi,\pi]^n$	100
f_9	-78.33236	$[-5,5]^n$	100
f_{10}	0	$[-5,10]^n$	100

3.2 Parameter Setting

We set the parameter values as follows:

- The size of population :

$$N=\begin{cases} 101 & n=100 \\ 31 & n=30 \end{cases}$$

- We use the good lattice point method [7] to construct the uniform arrays $U_{101}(101^{100})$ and $U_{31}(31^{30})$, which is used in Initialization.

- The number of solutions generated by in Reproduction at each generation $K = 12$, the number of solutions undergoing UOBQDA $J = 2$. The maximal number of the function value evaluations in the simplex method $S = [1.5n]$.
- In Reproduction Step, the $M = [N/2]$ best solutions from the current population are selected to construct the probability model, $H = 100$. is used in the histogram model.
- In the simplex method, the initial vertices $v^1, v^2, \cdots, v^{n+1}$ is generated in the following way:
  $$v^1 = v,$$
  and
  $$v^i = v + \lambda e^i, i = 2, 3, \cdots, n + 1,$$
  where e^i $(i = 1, 2, \cdots, n)$ are n unit vectors, and λ is a constant, we set $\lambda = 0.01$ in our experiments.
- In UOBDQA, we set $\rho_{beg} = 0.01$ and $\rho_{end} = 10^{-8}$.
- Stopping condition: When the current best function value can not be further reduced in 5 successive generations after 30 generations, then the execution of the algorithm is stopped.

  In our experimental study, we perform 30 independent runs for each algorithm on each test problem and record the mean number of function evaluations and the mean of the smallest function values found in 30 runs.

3.3 Effectiveness of the Components in the Algorithm

The main features of our algorithm are uniform design, utilization of two different local search techniques and EDA reproduction. Therefore, we would like to address the following three questions:

1. Does the uniform design in Initialization have any benefit?
2. Is it necessary to use two local search techniques in EDA/L? and
3. Is EDA reproduction operator better than commonly-used crossovers?

To investigate the above three questions, we compare EDA/L with the following three algorithms:

Algorithm A: It is the same as EDA/L, except randomly generates N solutions $\widetilde{x}^1, \widetilde{x}^2, \cdots, \widetilde{x}^N$ from D in Initialization.

Algorithm B: It is the same as EDA/L, except it does not apply the simplex method to the new solutions generated in Initialization and Reproduction, i.e., it simply lets $x^i = \widetilde{x}^i$ in Initialization and Reproduction.

Algorithm C: It is the same as EDA/L, except it uses crossovers instead of the EDA operator to generate new solutions in Reproduction. We have implemented three commonly used crossovers: one point crossover, uniform crossover, and intermediate crossover. Let $a = (a_1, a_2, \cdots, a_n)$ and $b = (b_1, b_2, \cdots, b_n)$ be the two parents, the one point crossover randomly pick up a crossover position $j \in \{2, 3, \cdots, n\}$, and the resultant offspring is $c = (a_1, \cdots, a_{j-1}, b_j, \cdots b_n)$. The resultant offspring of the uniform crossover is $c = (c_1, c_2, \cdots, c_n)$ where c_i is randomly picked up from $\{a_i, b_i\}$. In intermediate crossover, the offspring is $c = (\frac{a_1+b_1}{2}, \cdots, \frac{a_n+b_n}{2})$. In Reproduction in Algorithm C, we randomly pick up two distinct solutions from the current population, then apply a crossover to them to generate a new solution. Repeating this procedure K times will generate K new solutions.

We use f_2, f_7 and f_{10} as the test functions to compare the performances of these three algorithms and our proposed algorithm. The results are listed in Table 2 - 4.

Table 2: The effectiveness of the main components of the algorithm on f_2.

Algorithm	Mean No. of function Evaluations	Mean of the smallest function values found
EDA/L	75014	0
Algorithm A	77870	0.198992
Alogrithm B	72718	2.026529
Algorithm C with one point crossover	81706	0.021804
Algorithm C with uniform crossover	88141	0.0014333
Algorithm C with intermediate crossover	77477	2.586894

Table 3: The effectiveness of the main components of the algorithm on f_7.

Algorithm	Mean No. of function Evaluations	Mean of the smallest function values found
EDA/L	169887	-94.3757
Algorithm A	183891	-87.1928
Alogrithm B	179810	-80.2910
Algorithm C with one point crossover	189103	-83.4810
Algorithm C with uniform crossover	198381	-90.3381
Algorithm C with intermediate crossover	183491	-90.2874

Table 4: The effectiveness of the main components of the algorithm on f_{10}.

Algorithm	Mean No. of function Evaluations	Mean of the smallest function values found
EDA/L	1281401	4.324×10^{-3}
Algorithm A	1294983	5.981×10^{-2}
Alogrithm B	1592820	1.395×10^{-1}
Algorithm C with one point crossover	1509894	12.987412
Algorithm C with uniform crossover	1300304	1.292349
Algorithm C with intermediate crossover	1582901	2.183849

Since all the algorithms adopted the same framework and stopping condition, it is hardly surprising that all these algorithms require about the same number of function evaluations. EDA/L outperforms other algorithms in terms of solution quality. These results indicate that we can have positive answers to the three questions we asked earlier.

3.4 Performance of EDA/L

We have tested EDA/L on $f_1 - f_{10}$. Table 5 shows the experimental results.

Table 5: The performance of EDA/L

Test Function	Mean number of function evaluations	Mean of the smallest function values found	Globally minimal function value
f_1	52216	-12569.48	-12569.5
f_2	75014	0	0
f_3	106061	4.141×10^{-15}	0
f_4	79096	0	0
f_5	89925	3.654×10^{-21}	0
f_6	114570	3.485×10^{-21}	0
f_7	169887	-94.3757	-99.2784
f_8	124417	3.294×10^{-8}	0
f_9	153116	-78.31077	-78.33236
f_{10}	128140	4.324×10^{-3}	0

We can see that the means of the smallest function value found are equal or close to the globally minimal function values. At the first glance, one may judge that the performance of EDA/L is poor on f_7 . The mean of the smallest function found to f_7 is about 4.9% above the global minimal function value. However, there are 100! local minima for this function in its feasible region. Therefore it is extremely hard to find the global minimum. To our best knowledge, only the orthogonal genetic algorithm was tested on f_7 with 100 variables [10]. In the following comparison, we will see that our algorithm is better than the orthogonal genetic algorithm on this problem.

3.5 Comparison with other algorithms

The functions $f_1 - f_{10}$ have been tested by the following evolutionary algorithms recently:

- Orthogonal genetic algorithm with quantization (OGA/Q) [10]. This algorithm applies the orthogonal design to improve the performance of the genetic algorithm for global optimization. $f_1 - f_{10}$ and some other unimodel functions was tested by OGA/Q.
- Fast evolution strategy (FSA) [19]. This algorithm uses Cauchy mutation instead of Gaussian mutation in generating new test points. $f_1 - f_6$ was tested by FSA.
- Particle Swarm Optimization (PSO) [1]. It is a algorithm inspired by bird flocking. f_2, f_4,and f_{10} was tested by PSO.
- Evolution Programming With Gaussian Mutation (EP/GM) [6]. f_2, f_3, f_4, f_{10} was tested by EP/GM.

Tables 6 compares the quality of the solutions found by EDA/L with the above four algorithms, while table 7 compares the computational costs.

Table 6: Comparison of the mean of the smallest function value found

Test functions	EDA/L	OGA/Q	FSA	PSO	EP/GM
f_1	-12569.4811	-12569.4537	-12554.5	N/A	N/A
f_2	0	0	4.6×10^{-2}	46.4689	120.00
f_3	4.141×10^{-15}	4.440×10^{-16}	1.8×10^{-2}	N/A	9.10
f_4	0	0	1.6×10^{-2}	0.4498	2.52×10^{-7}
f_5	3.654×10^{-21}	6.019×10^{-6}	9.2×10^{-6}	N/A	N/A
f_6	3.485×10^{-21}	1.869×10^{-4}	1.6×10^{-4}	N/A	N/A
f_7	-94.3757	-92.83	N/A	N/A	N/A
f_8	3.294×10^{-8}	4.672×10^{-7}	N/A	N/A	N/A
f_9	-78.31077	-78.3000	N/A	N/A	N/A
f_{10}	4.324×10^{-3}	7.520×10^{-1}	N/A	1911.598	82.70

Table 7: Comparison of the mean no of the function evaluations

Test functions	EDA/L	OGA/Q	FSA	PSO	EP/GM
f_1	52,216	302,166	900,030	N/A	N/A
f_2	75,014	224,710	500,030	250,000	250,000
f_3	106,061	114,421	150,030	N/A	150,000
f_4	79,096	134,000	200,030	250,000	250,000
f_5	89,925	134,556	150,030	N/A	N/A
f_6	114,570	134,143	150,030	N/A	N/A
f_7	169.887	302,773	N/A	N/A	N/A
f_8	124.417	190,031	N/A	N/A	N/A
f_9	153,116	245,930	N/A	N/A	N/A
f_{10}	128,140	167,863	N/A	250,000	250,000

We can see that EDA/L can give better solutions than other algorithms on all the test functions except f_3 in which the solution obtained by OGA/Q is slightly better than EDA/L. We also see that the computational cost of our algorithm is the smallest on all the test problems. For f_5 as an example, EDA/L gives a mean of the smallest function values of 3.654×10^{-21} which is much more accurate than the solutions obtained by OGA/Q and FSA, but the mean number of the function evaluations is about 67% of that of OGA/Q, and about 60% of that of FSA. These results show that EDA/L can give a better solution at a lower computational cost than the existing algorithms. The C++ code of EDA/L can be obtained from the authors at qzhang@essex.ac.uk.

4 Conclusions

In this paper, we have introduced a hybrid estimation of distribution algorithm, called EDA/L, for global optimizations. The main features of the EDA/L are as follows:

- Initial points are generated by uniform design technique. Therefore, these points are evenly scattered over the feasible region.

- Two local search techniques are used in EDA/L. The incomplete simplex method is applied to every new point generated by uniform design and by EDA reproduction operator. UOBDQA applies to some selected resultant points of the incomplete simplex method. The incomplete simplex method is to help to determine promising points promising while UOBDQA locates the local minima.

- Our EDA reproduction operator is based on the histogram model, which is easy to build and sample. Although the modelling ability of the histogram model is limited, the local search techniques in EDA/L can compensate it.

The empirical studies show that the uniform design and utilization of two local searches are beneficial to the algorithm. We also show that our EDA reproduction operator is better than the three commonly-used crossovers. We tested EDA/L on 10 well-known benchmark problems. The results show that EDA/L are significant better than four existing evolutionary algorithms.

References

[1] Angeline, P.J. (1998), "Evolutionary optimization versus particle swarm optimization: Philosophy and performance differences," in *Proceedings of Evolutionary Programming*, VII, V.W. Porto, N.Saravanan, D.Waagen, and A.E.Eiben, (Eds). Berlin, Germany: Springer-Verlag, pp. 601-610.

[2] Baluja, S. and Davies, S (1997). "Combing Multiple Optimization Runs with Optimal Dependency Trees", Carnegie Mellon University, CMU-CS-97-157.

[3] Bosman, P. A. N. and Thierens, D. (2001), "Exploiting gradient information in continuous iterated density estimation evolutionary algorithms", working report, UU-CS-2001-53, Universiteit Utrecht.

[4] Bosman, P. A. N. and Thierens, D. (2000), "Expanding from Discrete to Continuous EDAs: The IEDA, *Proceedings of Parallel Problem Solving from Nature*, PPSN-VI, pp. 767-776

[5] Mūehlenbein, H. and Paaβ, G. (1996), "From Recombination of Genes to the Estimation of Distribution Part 1, Binary Parameter", Lecture Notes in Computer Science 1141, *Parallel Problem Solving from Nature*, pp. 178-187.

[6] Chellapilla, K. (1998), "Combining mutation operators in evolutionary programming", *IEEE Transactions on Evolutionary Computation*, vol. 2, pp. 91-96.

[7] Fang, K. T. and Wang, Y. (1994), *Number-theoretic methods in Statistics*, Chapman and Hall, London.

[8] Fletcher, R. (1987), *Practical Methods of Optimization,* John Wiley and Sons.

[9] Larrañaga, P. and Lozano, J. A. (2001), *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*, Kluwer Academic Publishers.

[10] Leung, Y. W. and Wang, Y. (2001), "An orthogonal genetic algorithm with quantization for global numerical optimization", *IEEE Transactions on Evolutionary Computation*, vol. 5, pp. 41-51.

[11] Merz, P. and Freisleben, B. (2000), "Fitness Landscape Analysis and Memetic Algorithms for the Quadratic Assignment Problem", *IEEE Transactions on Evolutionary Computation*, vol. 4, pp. 337-352.

[12] Montgomery, D. C. (1997), *Design and Analysis of Experiments*, 5th Edition, Wiley.

[13] Nelder, J. A. and Mead, R. (1965), "A simplex method for function minimization", *Computer Journal*, vol. 7, pp. 308-313.

[14] Pelikan, M. and Goldberg, D. E. and Cantú-Paz, E. (1999), "BOA: The Bayesian optimization algorithm", In *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-99*, W. Banzhaf and J. Daida and A. E. Eiben and M. H. Garzon and V. Honavar and M. Jakiela and R. E. Smith (Eds), Morgan Kaufmann Publishers, San Francisco, CA,Orlando, FL, vol. 1, pp. 525–532.

[15] Powell, M. J. D. (2002), "On trust region methods for unconstrained minimization without derivatives", working report, DAMTP 2002/NA02, Department of Applied Mathematics and Theoretical Physics, University of Cambridge.

[16] Robles, V. and de Miguel, P. and Larrañaga, P. (2001). "Solving the travelling salesman problem with Estimation of Distribution Algorithms", in P. Larrañaga and J. A. Lozano (Eds), *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*, Kluwer Academic Publishers.

[17] Torn, A. and Zilinskas, A. (1989), *Global Optimization,* Springer-Verlag.

[18] Tsutsui, S., Pelikan, M. and Goldberg, D. E. (2001), "Evolutionary Algorithm Using Marginal Histogram Models in Continuous Domain", IlliGAL Report 2001019, University of Illinois at Urbana-Champaign.

[19] Yao, X. and Liu, Y. (1999), “Evolutionary Programming Made Faster”, *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 2, pp. 82-102.

[20] Zhang, B. T. (1999), “A Bayesian Framework for Evolutionary Computation”, *Proceedings of the 1999 Congress on Evolutionary Computation*, vol.1, pp. 722-228.

[21] Zhang, Q. and Leung, Y. W. (1999), “An Orthogonal genetic algorithm for multimedia multicast routing”, *IEEE Transactions on Evolutionary Computation*, vol. 3, pp. 53-62.