Brief Papers

An Orthogonal Genetic Algorithm with Quantization for Global Numerical Optimization

Yiu-Wing Leung, *Senior Member, IEEE* and Yuping Wang

***Abstract*—We design a genetic algorithm called the *orthogonal genetic algorithm with quantization* for global numerical optimization with continuous variables. Our objective is to apply methods of experimental design to enhance the genetic algorithm, so that the resulting algorithm can be more robust and statistically sound. A quantization technique is proposed to complement an experimental design method called *orthogonal design*. We apply the resulting methodology to generate an initial population of points that are scattered uniformly over the feasible solution space, so that the algorithm can evenly scan the feasible solution space once to locate good points for further exploration in subsequent iterations. In addition, we apply the quantization technique and orthogonal design to tailor a new crossover operator, such that this crossover operator can generate a small, but representative sample of points as the potential offspring. We execute the proposed algorithm to solve 15 benchmark problems with 30 or 100 dimensions and very large numbers of local minima. The results show that the proposed algorithm can find optimal or close-to-optimal solutions.**

***Index Terms*—Evolutionary computation, experimental design methods, genetic algorithms, numerical algorithms, numerical optimization, orthogonal array, orthogonal design.**

I. Introduction

GLOBAL optimization problems arise in almost every field of science, engineering, and business. Many of these problems cannot be solved analytically, and consequently, they have to be addressed by numerical algorithms. In global optimization problems, the major challenge is that an algorithm may be trapped in the local optima of the objective function. This issue is particularly challenging when the dimension is high and there are numerous local optima. In fact, few researchers have tested their optimization algorithms on problems with 30 or more dimensions (e.g., see [1]–[7]).

Recently, Leung and Zhang [8] observed that some major steps of a genetic algorithm (GA) can be considered to be "experiments." For example, a crossover operator samples the genes from the parents to produce some potential offspring, and this operation can be considered to be a sampling experiment. They proposed to incorporate *experimental design methods* [9], [10] into the GA, so that the resulting algorithm can be more robust and statistically sound. In [11], they applied an experimental design method called *orthogonal design* to enhance the crossover operator for a zero–one integer programming problem, and they called the resulting algorithm, the *orthogonal genetic algorithm* (OGA). Numerical results demonstrated that the OGA had a significantly better performance than the traditional GA on the problems studied [11].

Manuscript received December 22, 1998; revised October 15, 1999 and April 10, 2000. This work was supported by the HKBU Faculty Research Grant FRG/97-98/II-08.

Y.-W. Leung is with the Department of Computer Science, Hong Kong Baptist University, Kowloon Tong, Hong Kong (e-mail: ywleung@comp.hkbu.edu.hk).

Y. Wang is with the Department of Applied Mathematics, Xidian University, Xi'an 710071, R.O.C.

Publisher Item Identifier S 1089-778X(01)00999-7.

Orthogonal design is applicable to discrete variables, but it is not applicable to continuous variables [9], [10]. In [11], the zero–one integer programming problem only involves discrete variables, and hence the orthogonal design can be applied to enhance the crossover operator for this problem. However, the orthogonal design is not directly applicable to enhance the genetic algorithm for optimization with continuous variables.

In this paper, we design a genetic algorithm called the *orthogonal genetic algorithm with quantization* (OGA/Q) for global numerical optimization with continuous variables. We propose a quantization technique to complement the orthogonal design, so that we can apply the resulting methodology to enhance the genetic algorithm for optimization with continuous variables. In particular, we propose the following two enhancements.

1) Before solving an optimization problem, we usually have no information about the location of the global minimum. It is desirable that an algorithm starts to explore those points that are scattered evenly in the feasible solution space. In this manner, the algorithm can evenly scan the feasible solution space once to locate good points for further exploration in subsequent iterations. As the algorithm iterates and improves the population of points, some points may move closer to the global minimum. We apply the quantization technique and the orthogonal design to generate this initial population.
2) We integrate the quantization technique and orthogonal design into the crossover operator, so that the resulting crossover operator can generate a small, but representative sample of points as the potential offspring.

We investigate the effectiveness of the proposed algorithm by solving 15 test functions with 30 or 100 dimensions.

II. Preliminary

A. Problem Definition

We consider the following global optimization problem:

$$\begin{aligned} &\text{minimize} && f(\boldsymbol{x}) \\ &\text{subject to} && \boldsymbol{l} \le \boldsymbol{x} \le \boldsymbol{u} \end{aligned}$$

TABLE I
EXPERIMENTAL DESIGN PROBLEM WITH THREE FACTORS AND THREE LEVELS PER FACTOR

	Factors		
	Temperature	Amount of fertiliers	pH value
Levels	20 °C	100 g/m^2	6
	25 °C	150 g/m^2	7
	30 °C	200 g/m^2	8

where $\boldsymbol{x} = (x_1, x_2, \ldots, x_N)$ is a variable vector in $\Re^N$, $f(\boldsymbol{x})$ is the objective function, and $\boldsymbol{l} = (l_1, l_2, \ldots, l_N)$ and $\boldsymbol{u} = (u_1, u_2, \ldots, u_N)$ define the feasible solution space. We denote the *domain* of x_i by $[l_i, u_i]$, and the feasible solution space by $[\boldsymbol{l}, \boldsymbol{u}]$.

B. Experimental Design Methods

We use an example to introduce the basic concept of experimental design methods. For more details, see [9] and [10]. The yield of a vegetable depends on: 1) the temperature, 2) the amount of fertilizer, and 3) the pH value of the soil. These three quantities are called the *factors* of the experiment. Each factor has three possible values shown in Table I, and we say that each factor has three *levels*.

To find the best combination of levels for a maximum yield, we can do one experiment for each combination, and then select the best one. In the above example, there are $3 \times 3 \times 3 = 27$ combinations, and hence there are 27 experiments. In general, when there are N factors and Q levels per factor, there are Q^N combinations. When N and Q are large, it may not be possible to do all Q^N experiments. Therefore, it is desirable to sample a small, but representative set of combinations for experimentation.

The *orthogonal design* was developed for this purpose [9], [10]. It provides a series of *orthogonal arrays* for different N and Q. We let $L_M(Q^N)$ be an orthogonal array for N factors and Q levels, where "L" denotes a Latin square and M is the number of combinations of levels. It has M rows, where every row represents a combination of levels. For convenience, we denote $L_M(Q^N) = [a_{i,j}]_{M \times N}$ where the jth factor in the ith combination has level $a_{i,j}$ and $a_{i,j} \in \{1, 2, \ldots, Q\}$. The following are two examples of orthogonal arrays:

$$L_4(2^3) = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 2 \\ 2 & 1 & 2 \\ 2 & 2 & 1 \end{bmatrix} \tag{1}$$

$$L_9(3^4) = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 2 & 2 \\ 1 & 3 & 3 & 3 \\ 2 & 1 & 2 & 3 \\ 2 & 2 & 3 & 1 \\ 2 & 3 & 1 & 2 \\ 3 & 1 & 3 & 2 \\ 3 & 2 & 1 & 3 \\ 3 & 3 & 2 & 1 \end{bmatrix}. \tag{2}$$

TABLE II
BASED ON THE ORTHOGONAL ARRAY $L_9(3^4)$, NINE REPRESENTATIVE COMBINATIONS ARE SELECTED FOR EXPERIMENTATION

Combination	Factor		
	Temperature	Amount of fertilizer	pH value
1st	20°C	100 g/m^2	6
2nd	20°C	150 g/m^2	7
3rd	20°C	200 g/m^2	8
4th	25°C	100 g/m^2	7
5th	25°C	150 g/m^2	8
6th	25°C	200 g/m^2	6
7th	30°C	100 g/m^2	8
8th	30°C	150 g/m^2	6
9th	30°C	200 g/m^2	7

In $L_4(2^3)$, there are three factors, two levels per factor, and four combinations of levels. In the first combination, the three factors have respective levels 1, 1, 1; in the second combination, the three factors have respective levels 1, 2, 2, etc. Similarly, in $L_9(3^4)$, there are four factors, three levels per factor, and nine combinations of levels.

In the above example, there are 27 combinations to be tested. We apply the orthogonal array $L_9(3^4)$ to select nine representative combinations to be tested, and these nine combinations are shown in Table II.

In general, the orthogonal array $L_M(Q^N)$ has the following properties.

1) For the factor in any column, every level occurs M/Q times.
2) For the two factors in any two columns, every combination of two levels occurs M/Q^2 times.
3) For the two factors in any two columns, the M combinations contain the following combinations of levels: $(1, 1), (1, 2), \cdots, (1, Q), (2, 1), (2, 2), \cdots, (2, Q), \cdots, (Q, 1), (Q, 2), \cdots, (Q, Q)$.
4) If any two columns of an orthogonal array are swapped, the resulting array is still an orthogonal array.
5) If some columns are taken away from an orthogonal array, the resulting array is still an orthogonal array with a smaller number of factors.

Consequently, the selected combinations are scattered uniformly over the space of all possible combinations. Fig. 1 shows an example. Orthogonal design has been proven optimal for additive and quadratic models, and the selected combinations are good representatives for all of the possible combinations [12].

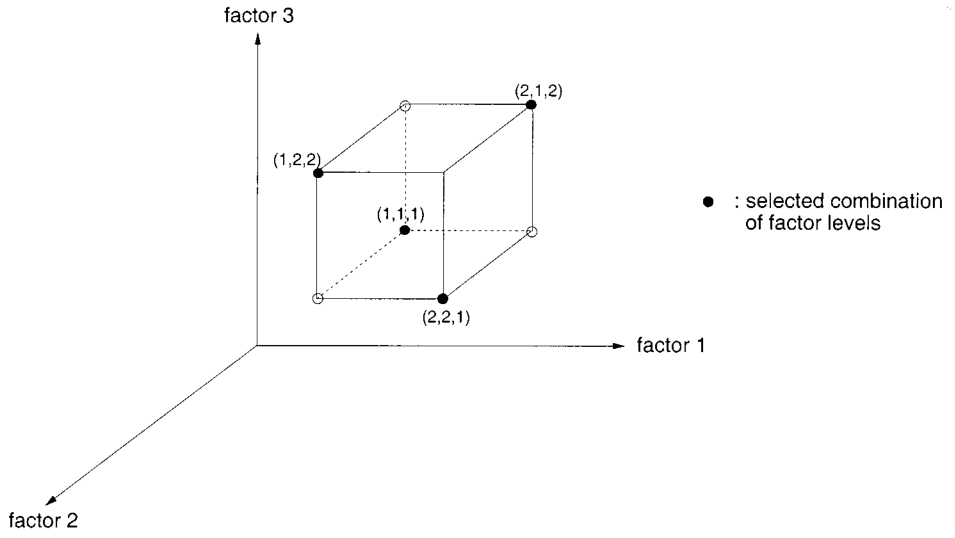

Fig. 1. Orthogonality of the orthogonal array $L_4(2^3)$.

C. Construction of Orthogonal Array

As we will explain, the proposed algorithm may require different orthogonal arrays for different optimization problems. Although many orthogonal arrays have been tabulated in the literature (e.g., see [13]), it is impossible to store all of them for the proposed algorithm. We will only need a special class of orthogonal arrays $L_M(Q^N)$, where Q is odd and $M = Q^J$, where J is a positive integer fulfilling

$$N = \frac{Q^J - 1}{Q - 1}. \tag{3}$$

In this subsection, we design a simple permutation method to construct orthogonal arrays of this class.

We denote the jth column of the orthogonal array $[a_{i,j}]_{M \times N}$ by $\boldsymbol{a}_j$. Columns $\boldsymbol{a}_j$ for $j = 1, 2, (Q^2-1)/(Q-1)+1, (Q^3-1)/(Q-1)+1, \cdots, (Q^{J-1}-1)/(Q-1)+1$ are called the *basic columns*, and the others are called the *nonbasic columns*. We first construct the basic columns, and then construct the nonbasic columns. The details are as follows.

```
Algorithm 1: Construction of Orthogonal Array:
  Step 1: Construct the basic columns as follows:
    FOR $k = 1$ TO $J$ DO
    BEGIN
      $j = \frac{Q^{k-1}-1}{Q-1} + 1$;
      FOR $i = 1$ TO $Q^J$ DO
        $a_{i,j} = \left\lfloor \frac{i-1}{Q^{J-k}} \right\rfloor$ mod $Q$;
    END.
  Step 2: Construct the nonbasic columns as follows:
    FOR $k = 2$ TO $J$ DO
    BEGIN
      $j = \frac{Q^{k-1}-1}{Q-1} + 1$;
      FOR $s = 1$ TO $j - 1$ DO
      FOR $t = 1$ TO $Q - 1$ DO
      $\boldsymbol{a}_{j+(s-1)(Q-1)+t} = (\boldsymbol{a}_s \times t + \boldsymbol{a}_j)$ mod $Q$;
    END.
  Step 3: Increment $a_{i,j}$ by one for all
    $1 \le i \le M$ and $1 \le j \le N$.
```

Example 1: Suppose $N = 4$ and $Q = 3$, and we want to construct the orthogonal array $L_{3^2}(3^4)$. The execution of Algorithm 1 is as follows.

Step 1: Construct the basic columns $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$ as follows. When $k = 1$, j is found to be 1 and $\boldsymbol{a}_1$ can be found to be

$$\boldsymbol{a}_1 = [0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 1 \quad 2 \quad 2 \quad 2]^T.$$

When $k = 2$, j is found to be 2 and $\boldsymbol{a}_2$ can be found to be

$$\boldsymbol{a}_2 = [0 \quad 1 \quad 2 \quad 0 \quad 1 \quad 2 \quad 0 \quad 1 \quad 2]^T.$$

Step 2: Construct the nonbasic columns $\boldsymbol{a}_3$ and $\boldsymbol{a}_4$ as follows. Since $J = 2$, k is always equal to 2, and hence $j = 2$ and $s = 1$. When $t = 1$, $\boldsymbol{a}_3$ is given by

$$\begin{aligned}\boldsymbol{a}_3 &= (\boldsymbol{a}_1 \times 1 + \boldsymbol{a}_2) \bmod 3\\ &= [0 \quad 1 \quad 2 \quad 1 \quad 2 \quad 3 \quad 2 \quad 3 \quad 4]^T \bmod 3\\ &= [0 \quad 1 \quad 2 \quad 1 \quad 2 \quad 0 \quad 2 \quad 0 \quad 1]^T.\end{aligned}$$

When $t = 2$, $\boldsymbol{a}_4$ is given by

$$\begin{aligned}\boldsymbol{a}_4 &= (\boldsymbol{a}_1 \times 2 + \boldsymbol{a}_2) \bmod 3\\ &= [0 \quad 1 \quad 2 \quad 2 \quad 3 \quad 4 \quad 4 \quad 5 \quad 6]^T \bmod 3\\ &= [0 \quad 1 \quad 2 \quad 2 \quad 0 \quad 1 \quad 1 \quad 2 \quad 0]^T.\end{aligned}$$

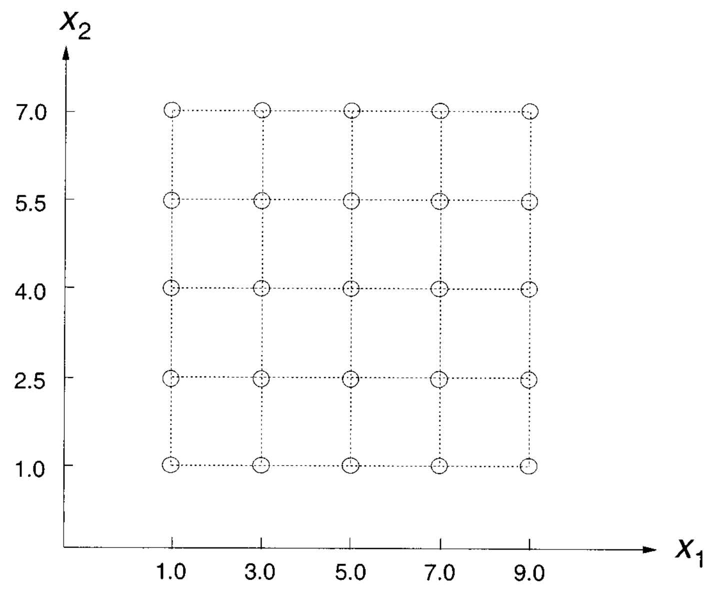

Fig. 2. Quantization for a two-dimensional optimization problem; the domains of x_1 and x_2 are [1.0, 9.0] and [1.0, 7.0] respectively. Each factor is quantized into five levels. After quantization, the feasible solution space contains 25 chromosomes.

Step 3: Increment $a_{i,j}$ for all $1 \le i \le 9$ and $1 \le j \le 4$. Concatenate $\boldsymbol{a}_1$, $\boldsymbol{a}_2$, $\boldsymbol{a}_3$, and $\boldsymbol{a}_4$ to form $L_{3^2}(3^4)$, which is shown in (2).

III. Orthogonal Genetic Algorithm with Quantization

We define $\boldsymbol{x} = (x_1, x_2, \ldots, x_N)$ to be a chromosome with cost $f(\boldsymbol{x})$. The optimization problem is equivalent to finding a chromosome of minimal cost.

A. Generation of Initial Population

Before an optimization problem is solved, we may have no information about the location of the global minimum. It is desirable that the chromosomes of the initial population be scattered uniformly over the feasible solution space, so that the algorithm can explore the whole solution space evenly.

We observe that an orthogonal array specifies a small number of combinations that are scattered uniformly over the space of all the possible combinations. Therefore, orthogonal design is a potential method for generating a good initial population.

We define x_i to be the ith factor, so that each chromosome has N factors. These factors are continuous, but the orthogonal design is applicable to discrete factors only. To overcome this issue, we quantize each factor into a finite number of values. In particular, we quantize the domain $[l_i, u_i]$ of x_i into Q_1 levels $\alpha_{i,1}, \alpha_{i,2}, \alpha_{i,3}, \ldots, \alpha_{i,Q}$, where the design parameter Q_1 is odd and $\alpha_{i,j}$ is given by

$$\alpha_{i,j} = \begin{cases} l_i & j = 1 \\ l_i + (j-1)\left(\dfrac{u_i - l_i}{Q_1 - 1}\right) & 2 \le j \le Q_1 - 1 \\ u_i & j = Q_1. \end{cases} \tag{4}$$

In other words, the difference between any two successive levels is the same. For convenience, we call $\alpha_{i,j}$ the jth level of the ith factor, and we denote $\alpha_i = (\alpha_{i,1}, \alpha_{i,2}, \ldots, \alpha_{i,Q_1})$. Fig. 2 shows an example of quantization.

After quantization, x_i has Q_1 possible values $\alpha_{i,1}, \alpha_{i,2}, \ldots, \alpha_{i,Q_1})$, and hence the feasible solution space contains Q_1^N points. We apply orthogonal design to select a small sample of points that are scattered uniformly over the feasible solution space.

We first construct a suitable orthogonal array. Recall that Algorithm 1 can only construct $L_{M_1}(Q_1^N)$, where $M_1 = Q_1^{J_1}$ and J_1 is a positive integer fulfilling $(Q_1^{J_1} - 1)/(Q_1 - 1) = N$ [see (3)]. Since the problem dimension N is given, there may not exist Q_1 and J_1 fulfilling this condition. We bypass this restriction as follows. We choose the smallest J_1 such that

$$\frac{Q_1^{J_1} - 1}{Q_1 - 1} \ge N. \tag{5}$$

We execute Algorithm 1 to construct an orthogonal array with $N' = (Q_1^{J_1} - 1)/(Q_1 - 1)$ factors, and then delete the last $N' - N$ columns to get an orthogonal array with N factors. The details are given in the following algorithm.

```
Algorithm 2: Construction of $L_{M_1}(Q_1^N)$:
  Step 1: Select the smallest $J_1$
           fulfilling $(Q_1^{J_1} - 1)/(Q_1 - 1) \ge N$.
  Step 2: If $(Q_1^{J_1} - 1)/(Q_1 - 1) = N$, then
           $N' = N$ else $N' = (Q_1^{J_1} - 1)/(Q_1 - 1)$.
  Step 3: Execute Algorithm 1 to construct
           the orthogonal array $L_{Q_1^{J_1}}(Q_1^{N'})$.
  Step 4: Delete the last $N' - N$ columns of
           $L_{Q_1^{J_1}}(Q_1^{N'})$ to get $L_{M_1}(Q_1^N)$ where
           $M_1 = Q_1^{J_1}$.
```

Example 2: We construct an orthogonal array with three factors and three levels per factor. The execution of Algorithm 2 is as follows.

Step 1: Since $N = 3$ and $Q_1 = 3$, J_1 is found to be 2.
Step 2: $N' = (3^2 - 1)/(3 - 1) = 4$.

Step 3: Execute Algorithm 1 to construct $L_{3^2}(3^4)$, and the details are given in Example 1.

Step 4: Delete the last $N' - N$ column of $L_{3^2}(3^4)$ to get $L_{3^2}(3^3)$:

$$L_{3^2}(3^3) = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 2 \\ 1 & 3 & 3 \\ 2 & 1 & 2 \\ 2 & 2 & 3 \\ 2 & 3 & 1 \\ 3 & 1 & 3 \\ 3 & 2 & 1 \\ 3 & 3 & 2 \end{bmatrix}. \tag{6}$$

After constructing $L_{M_1}(Q_1^N) = [a_{i,j}]_{M_1 \times N}$ we get a sample of M_1 combinations out of Q_1^N combinations. We apply these M_1 combinations to generate the following M_1 chromosomes:

$$\begin{cases} (\alpha_{1,a_{1,1}}, \alpha_{2,a_{1,2}}, \cdots, \alpha_{N,a_{1,N}}) \\ (\alpha_{1,a_{2,1}}, \alpha_{2,a_{2,2}}, \cdots, \alpha_{N,a_{2,N}}) \\ \cdots \\ (\alpha_{1,a_{M_1,1}}, \alpha_{2,a_{M_1,2}}, \cdots, \alpha_{N,a_{M_1,N}}). \end{cases} \tag{7}$$

Among these M_1 potential chromosomes, we select G chromosomes having the smallest cost as the initial population, where G is the population size. In this manner, we have evenly scanned the feasible solution space once to locate potentially good points for further exploration in subsequent iterations.

When the feasible solution space is large, it may be desirable to generate more potential chromosomes for a better coverage. However, the value of M_1 depends on N and Q_1, and it cannot be increased arbitrarily. To overcome this issue, we divide the feasible solution space into S subspaces, where S is a design parameter. In particular, we choose the sth dimension such that

$$u_s - l_s = \max_{1 \le i \le N} \{u_i - l_i\} \tag{8a}$$

and divide $[\boldsymbol{l}, \boldsymbol{u}]$ along the sth dimension into the following S subspaces $[\boldsymbol{l}(1), \boldsymbol{u}(1)], [\boldsymbol{l}(2), \boldsymbol{u}(2)], \ldots, [\boldsymbol{l}(S), \boldsymbol{u}(S)]$ where

$$\begin{cases} \boldsymbol{l}(i) = \boldsymbol{l} + (i-1)\left(\dfrac{u_s - l_s}{S}\right)\mathbf{1}_s \\ \boldsymbol{u}(i) = \boldsymbol{u} - (S-i)\left(\dfrac{u_s - l_s}{S}\right)\mathbf{1}_s \end{cases} \quad i = 1, 2, \ldots, S \tag{8b}$$

and $\mathbf{1}_s$ is an N-dimensional vector such that its sth element is one and all of the other elements are zero. We apply $L_{M_1}(Q_1^N)$ to each subspace to generate M_1 chromosomes, so that we get a total of $M_1 S$ potential chromosomes for the initial population. We select G chromosomes having the smallest cost as the initial population.

The details for generating an initial population are given as follows.

```
Algorithm 3: Generation of Initial Population:
  Step 1: Divide the feasible solution space $[\boldsymbol{l}, \boldsymbol{u}]$ into $S$ subspaces
          $[\boldsymbol{l}(1), \boldsymbol{u}(1)]$, $\boldsymbol{l}(2), \boldsymbol{u}(2)], \cdots, [\boldsymbol{l}(S), \boldsymbol{u}(S)]$
          based on (8).
  Step 2: Quantize each subspace based on (4), and then apply $L_{M_1}(Q_1^N)$ to
          select $M_1$ chromosomes based on (7).
  Step 3: Among the $M_1 S$ chromosomes, select $G$ chromosomes having the
          smallest cost as the initial population.
```

Example 3: Consider a three-dimensional optimization problem. Suppose $0.5 \le x_1 \le 10.5$, $3.5 \le x_2 \le 6.5$, and $4.5 \le x_3 \le 7.5$, and hence the feasible solution space $[\boldsymbol{l}, \boldsymbol{u}]$ is [(0.5, 3.5, 4.5), (10.5, 6.5, 7.5)]. We choose $Q_1 = 3$ and $S = 5$. The execution of Algorithm 3 is as follows.

Step 1: Divide the feasible solution space $[\boldsymbol{l}, \boldsymbol{u}]$ into the following five subspaces:

$$[\boldsymbol{l}(i), \boldsymbol{u}(i)] = [(2i - 1.5, 3.5, 4.5), (2i + 0.5, 6.5, 7.5)], \quad \text{for } 1 \le i \le 5.$$

Step 2: Quantize the subspace $[\boldsymbol{l}(1), \boldsymbol{u}(1)] = [(0.5, 3.5, 4.5), (2.5, 6.5, 7.5)]$ based on (4) to get

$$\begin{cases} \alpha_1 = (0.5, 1.5, 2.5) \\ \alpha_2 = (3.5, 5.0, 6.5) \\ \alpha_3 = (4.5, 6.0, 7.5) \end{cases}$$

and then apply $L_{3^2}(3^3)$ to select the following nine chromosomes based on (7):

$$\begin{cases} (0.5, 3.5, 4.5) \\ (0.5, 5.0, 6.0) \\ (0.5, 6.5, 7.5) \\ (1.5, 3.5, 6.0) \\ (1.5, 5.0, 7.5) \\ (1.5, 6.5, 4.5) \\ (2.5, 3.5, 7.5) \\ (2.5, 5.0, 4.5) \\ (2.5, 6.5, 6.0). \end{cases}$$

Fig. 3 shows the distribution of the nine selected chromosomes in the space $[\boldsymbol{l}(1), \boldsymbol{u}(1)]$. Proceed in a similar manner for the other four subspaces.

Step 3: Among the $9 \times 5 = 45$ potential chromosomes, select the G chromosomes having the smallest cost as the initial population.

p

B. Orthogonal Crossover with Quantization

We design a new crossover operator called *orthogonal crossover with quantization*. It acts on two parents. It quantizes the solution space defined by these parents into a finite number

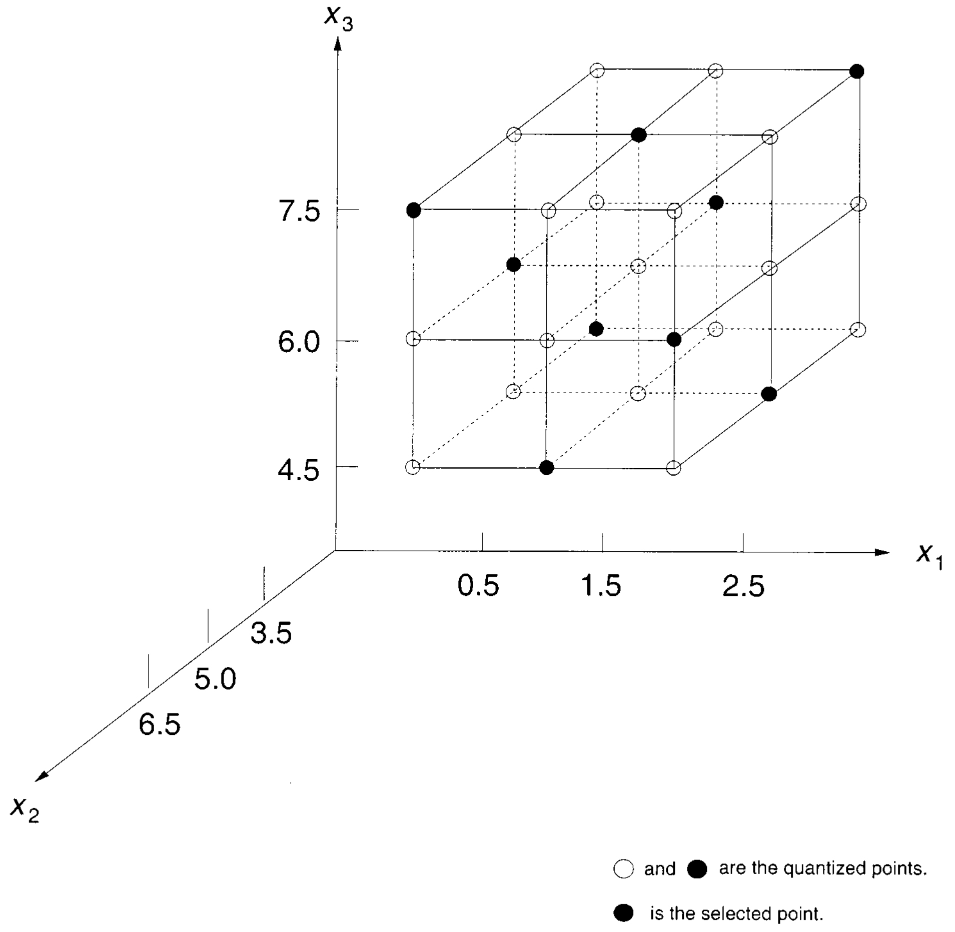

Fig. 3. Distribution of the nine selected chromosomes in the space $[\boldsymbol{l}(1), \boldsymbol{u}(1)]$ in Example 3.

of points, and then applies orthogonal design to select a small, but representative sample of points as the potential offspring.

Consider any two parents $\boldsymbol{p}_1 = (p_{1,1}, p_{1,2}, \ldots, p_{1,N})$ and $\boldsymbol{p}_2 = (p_{2,1}, p_{2,2}, \ldots, p_{2,N})$. They define the solution space $[\boldsymbol{l}_{\text{parent}}, \boldsymbol{u}_{\text{parent}}]$ where

$$\begin{cases} \boldsymbol{l}_{\text{parent}} = [\min(p_{1,1}, p_{2,1}), \min(p_{1,2}, p_{2,2}), \ldots, \\ \qquad\qquad \min(p_{1,N}, p_{2,N})] \\ \boldsymbol{u}_{\text{parent}} = [\max(p_{1,1}, p_{2,1}), \max(p_{1,2}, p_{2,2}), \ldots, \\ \qquad\qquad \max(p_{1,N}, p_{2,N})]. \end{cases} \tag{9}$$

We quantize each domain of $[\boldsymbol{l}_{\text{parent}}, \boldsymbol{u}_{\text{parent}}]$ into Q_2 levels such that the difference between any two successive levels is the same. Specifically, we quantize the domain of the ith dimension into $\beta_{i,1}, \beta_{i,2}, \ldots, \beta_{i,Q_2}$ where

$$\beta_{i,j} = \begin{cases} \min(p_{1,i}, p_{2,i}), & j = 1 \\ \min(p_{1,i}, p_{2,i}) + (j-1) \\ \quad \cdot \left(\dfrac{|p_{1,i} - p_{2,i}|}{Q_2 - 1}\right), & 2 \le j \le Q_2 - 1 \\ \max(p_{1,i}, p_{2,i}), & j = Q_2. \end{cases} \tag{10}$$

We denote $\beta_i = (\beta_{i,1}, \beta_{i,2}, \ldots, \beta_{i,Q_2})$. As the population is being evolved and improved, the population members are getting closer to each other, so that the solution space defined by two parents is becoming smaller. Since Q_2 is fixed, the quantized points are getting closer, and hence we can get increasingly precise results.

After quantizing $[\boldsymbol{l}_{\text{parent}}, \boldsymbol{u}_{\text{parent}}]$, we apply orthogonal design to select a small, but representative sample of points as the potential offspring, and then select the ones with the smallest cost to be the offspring. Each pair of parents should not produce too many potential offspring in order to avoid a large number of function evaluations during selection. For this purpose, we divide the variables $x_1, x_2, \ldots, x_N$ into F groups, where F is a small design parameter, and each group is treated as one factor. Consequently, the corresponding orthogonal array has a small number of combinations, and hence a small number of potential offspring are generated. Specifically, we randomly generate $F-1$ integers $k_1, k_2, \ldots, k_{F-1}$ such that $1 < k_1 < k_2 < \ldots < k_{F-1} < N$, and then create the following F factors for any chromosome $\boldsymbol{x} = (x_1, x_2, \ldots, x_N)$:

$$\begin{cases} \boldsymbol{f}_1 = (x_1, \ldots, x_{k_1}) \\ \boldsymbol{f}_2 = (x_{k_1+1}, \ldots, x_{k_2}) \\ \cdots \\ \boldsymbol{f}_F = (x_{k_{F-1}}+1, \ldots, x_N). \end{cases} \tag{11}$$

Since $x_1, x_2, \ldots, x_N$ have been quantized, we define the following Q_2 levels for the ith factor $\boldsymbol{f}_i$:

$$\begin{cases} \boldsymbol{f}_i(1) = (\beta_{k_{i-1}+1,1}, \beta_{k_{i-1}+2,1}, \ldots, \beta_{k_i,1}) \\ \boldsymbol{f}_i(2) = (\beta_{k_{i-1}+1,2}, \beta_{k_{i-1}+2,2}, \ldots, \beta_{k_i,2}) \\ \cdots \\ \boldsymbol{f}_i(Q_2) = (\beta_{k_{i-1}+1,Q_2}, \beta_{k_{i-1}+2,Q_2}, \ldots, \beta_{k_i,Q_2}). \end{cases} \tag{12}$$

There are a total of Q_2^F combinations of levels. We apply the orthogonal array $L_{M_2}(Q_2^F)$ to select a sample of M_2 chromosomes as the potential offspring. We remind that Algorithm 1 can only construct $L_{M_2}(Q_2^F)$ where Q_2 is odd and $M_2 = Q_2^{J_2}$ where J_2 is a positive integer fulfilling $(Q_2^{J_2}-1)/(Q_2-1) = F$ [see (3)]. We bypass this restriction in a manner similar to that in Algorithm 2. Specifically, we select the smallest J_2 such that $(Q_2^{J_2}-1)/(Q_2-1) \geq F$, then execute Algorithm 1 to construct an orthogonal array with $F' = (Q_2^{J_2}-1)/(Q_2-1)$ factors, and then delete the last $F'-F$ columns of this array. The resulting array has F factors only, and it is denoted by $L_{M_2}(Q_2^F) = [b_{i,j}]_{M_2 \times F}$. The details are given as follows.

```
Algorithm 4: Construction of $L_{M^2}(Q_2^F)$:
  Step 1: Select the smallest $J_2$
          fulfilling $(Q_2^{J_2}-1)/(Q_2-1) \geq F$.
  Step 2: If $(Q_2^{J_2}-1)/(Q_2-1) = F$, then
          $F' = F$ else $F' = (Q_{2^{J_2}}-1)/(Q_2-1)$.
  Step 3: Execute Algorithm 1 to construct
          the orthogonal array $L_{Q_2^{J_2}}(Q_{2^{F'}})$.
  Step 4: Delete the last $F'-F$ columns of
          $L_{Q_2^{J_2}}(Q_2^{F'})$ to get $L_{M_2}(Q_2^F)$.
```

We apply $L_{M_2}(Q_2^F)$ to generate the following M_2 chromosomes out of Q_2^F possible chromosomes:

$$\begin{cases} (\boldsymbol{f}_1(b_{1,1}), \boldsymbol{f}_2(b_{1,2}), \ldots, \boldsymbol{f}_F(b_{1,F})) \\ (\boldsymbol{f}_1(b_{2,1}), \boldsymbol{f}_2(b_{2,2}), \ldots, \boldsymbol{f}_F(b_{2,F})) \\ \cdots \\ (\boldsymbol{f}_1(b_{M_2,1}), \boldsymbol{f}_2(b_{M_2,2}), \ldots, \boldsymbol{f}_F(b_{M_2,F})). \end{cases} \tag{13}$$

The details of orthogonal crossover with quantization are given as follows.

```
Algorithm 5: Orthogonal Crossover with
Quantization:
  Step 1: Quantize $[\boldsymbol{l}_{\text{parent}}, \boldsymbol{u}_{\text{parent}}]$ based on
          (10).
  Step 2: Randomly generate $F-1$ integers
          $k_1, k_2, \ldots, k_{F-1}$ such that
          $1 < k_1 < k_2 < \cdots < k_{F-1} < N$.
          Create $F$ factors based on (11).
  Step 3: Apply $L_{M_2}(Q_2^F)$ to generate $M_2$
          potential offspring based on
          (13).
```

Example 4: Consider a five-dimensional optimization problem. Let the two parents be $\boldsymbol{p}_1 = (0, 4, 2, 0, 1)$ and $\boldsymbol{p}_2 = (6, 1, 5, -3, 2)$. These parents define the solution space $[\boldsymbol{l}_{\text{parent}}, \boldsymbol{u}_{\text{parent}}] = [(0, 1, 2, -3, 1), (6, 4, 5, 0, 2)]$. We choose $F = 4$ and $Q_2 = 3$. The execution of Algorithm 5 is as follows.

Step 1: Quantize $[\boldsymbol{l}_{\text{parent}}, \boldsymbol{u}_{\text{parent}}]$ into

$$\begin{cases} \beta_1 = (0.0, 3.0, 6.0) \\ \beta_2 = (1.0, 2.5, 40.) \\ \beta_3 = (2.0, 3.5, 5.0) \\ \beta_4 = (-3.0, -1.50, 0.0) \\ \beta_5 = (1.0, 1.5, 2.0). \end{cases}$$

Step 2: Suppose $k_1 = 2$, $k_2 = 3$, $k_3 = 4$, and $k_4 = 5$. Create the following four factors: $\boldsymbol{f}_1 = (x_1, x_2)$, $\boldsymbol{f}_2 = (x_3)$, $\boldsymbol{f}_3 = (x_4)$, $\boldsymbol{f}_4 = (x_5)$.

Step 3: Apply $L_{3^2}(3^4)$ to get the following nine potential offspring:

$$\begin{cases} (0.0, 1.0, 2.0, -3.0, 1.0) \\ (0.0, 1.0, 3.5, -1.5, 1.5) \\ (0.0, 1.0, 5.0, 0.0, 2.0) \\ (3.0, 2.5, 2.0, -1.5, 2.0) \\ (3.0, 2.5, 3.5, 0.0, 1.0) \\ (3.0, 2.5, 5.0, -3.0, 1.5) \\ (6.0, 4.0, 2.0, 0.0, 1.5) \\ (6.0, 4.0, 3.5, -3.0, 2.0) \\ (6.0, 4.0, 5.0, -1.5, 1.0). \end{cases}$$

C. Orthogonal Genetic Algorithm with Quantization

We generate a good initial population with G chromosomes, and then evolve and improve the population iteratively. In each iteration, we apply orthogonal crossover with quantization and mutation to generate a set of potential offspring. Among these potential offspring and the parents, we select the G chromosomes with the least cost to form the next generation. We let $\boldsymbol{P}_{\text{gen}}$ be a population of chromosomes in the ith generation. The details of the overall algorithm are as follows.

```
Orthogonal Genetic Algorithm with
Quantization:
  Step 1:   **Initialization**
  Step 1.1: Execute Algorithm 2 to
            construct $L_{M_1}(Q_1^N)$.
  Step 1.2: Execute Algorithm 3 to
            generate an initial population
            $\boldsymbol{P}_0$. Initialize the generation
            number gen to 0.
  Step 1.3: Execute Algorithm 4 to
            construct $L_{M_2}(Q_2^F)$.
  Step 2:   **Population Evolution**
            WHILE (stopping condition is
            not met) DO
            BEGIN
  Step 2.1: **Orthogonal Crossover with
            Quantization**—Each chromosome
            in $\boldsymbol{P}_{\text{gen}}$ is selected for
            crossover with probability $p_c$.
```

```
               When the number of selected
               chromosomes is odd, select
               one additional chromosome
               from $P_{\text{gen}}$ randomly. Let the
               number of selected chromosomes
               be $2i$. Form $i$ random pairs of
               chromosomes. Execute Algorithm
               5 to perform orthogonal
               crossover with quantization on
               each pair.
Step 2.2:      Mutation—Each chromosome in
               $P_{\text{gen}}$ undergoes mutation with
               probability $p_m$. To perform
               mutation on a chromosome,
               randomly generate an integer
               $j \in [1, N]$ and a real number
               $z \in [l_j, u_j]$, and then replace the
               $j$th component of the chosen
               chromosome by $z$ to get a new
               chromosome.
Step 2.3:      Selection—Among the
               chromosomes in $P_{\text{gen}}$ and those
               generated by crossover and
               mutation, select the $G$
               chromosomes with the least
               cost to form the next
               generation $P_{\text{gen}+1}$.
Step 2.4:      Increment the generation
               number gen by 1.
             END
```

In Step 2, the population is evolved and improved iteratively until a stopping condition is met. Similar to the other genetic algorithms, there can be many possible stopping conditions. For example, one possible stopping condition is to stop when the best chromosome cannot be further improved in a certain number of generations.

IV. Numerical Experiments and Results

A. Test Functions

We execute the OGA/Q to solve the following test functions:

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

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

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

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

$$f_5 = \frac{\pi}{N}\left\{10\sin^2(\pi y_i) + \sum_{i=1}^{N-1}(y_i - 1)^2 \cdot [1 + 10\sin^2(\pi y_{i+1})] + (y_N - 1)^2\right\} + \sum_{i=1}^{N} u(x_i, 10, 100, 4)$$

where

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

and

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

$$f_6 = \frac{1}{10}\left\{\sin^2(3\pi x_1) + \sum_{i=1}^{N-1}(x_i - 1)^2[1 + \sin^2(3\pi x_{i+1})] + (x_N - 1)^2[1 + \sin^2(2\pi x_N)]\right\} + \sum_{i=1}^{N} u(x_i, 5, 100, 4)$$

$$f_7 = -\sum_{i=1}^{N}\sin(x_i)\sin^{20}\left(\frac{i \times x_i^2}{\pi}\right)$$

$$f_8 = \sum_{i=1}^{N}\left[\sum_{j=1}^{N}(\chi_{ij}\sin\omega_j + \psi_{ij}\cos\omega_j) - \sum_{j=1}^{N}(\chi_{ij}\sin x_j + \psi_{ij}\cos x_j)\right]^2$$

where χ_{ij} and ψ_{ij} are random integers in $[-100, 100]$, and ω_j is a random number in $[-\pi, \pi]$.

$$f_9 = \frac{1}{N}\sum_{i=1}^{N}(x_i^4 - 16x_i^2 + 5x_i)$$

$$f_{10} = \sum_{j=1}^{N-1}\left[100(x_j - x_{j+1})^2 + (x_j - 1)^2\right]$$

$$f_{11} = \sum_{i=1}^{N} x_i^2$$

$$f_{12} = \sum_{i=1}^{N} x_i^4 + \text{random}[0, 1)$$

$$f_{13} = \sum_{i=1}^{N}|x_i| + \prod_{i=1}^{N}|x_i|$$

$$f_{14} = \sum_{i=1}^{N}\left(\sum_{j=1}^{i} x_j\right)^2$$

$$f_{15} = \max\{|x_i|, \quad i = 1, 2, \cdots, N\}.$$

Table III lists the basic characteristics of these test functions.

TABLE III
BASIC CHARACTERISTICS OF THE TEST FUNCTIONS

Test function	Feasible solution space	Globally minimal function value	Number of local minima
f_1	$[-500, 500]^N$	-12569.5	NA
f_2	$[-5.12, 5.12]^N$	0	NA
f_3	$[-32, 32]^N$	0	NA
f_4	$[-600, 600]^N$	0	NA
f_5	$[-50, 50]^N$	0	NA
f_6	$[-50, 50]^N$	0	NA
f_7	$[0, \pi]^N$	–99.2784	$N!$
f_8	$[-\pi, \pi]^N$	0	2^N
f_9	$[-5, 5]^N$	-78.33236	2^N
f_{10}	$[-5, 10]^N$	0	NA
f_{11}	$[-100, 100]^N$	0	NA
f_{12}	$[-1.28, 1.28]^N$	0	NA
f_{13}	$[-10, 10]^N$	0	NA
f_{14}	$[-100, 100]^N$	0	NA
f_{15}	$[-100, 100]^N$	0	NA

The above test functions were examined in [3]–[5] and [14]–[18] and Table IV lists the problem dimensions adopted in these studies. In our study, we execute OGA/Q to solve these test functions with the following dimensions.

- The problem dimension for f_1–f_6 is 30, and the problem dimension for f_7–f_{10} is 100. In this manner, these test functions have so many local minima that they are challenging enough for performance evaluation, and the existing results reported in [4], [5] can be used for a direct comparison.
- The problem dimension for f_{11}–f_{15} is 30. In this manner, the existing results reported in [17], [18] can be used for a direct comparison.

B. Existing Algorithms for Comparison

In some recent studies, f_1–f_{15} were tested by the following optimization algorithms:

1) *Fast evolution strategy (FES)* [5]: FES uses evolution strategies with Cauchy mutation to generate offspring for each new generation.
2) *Enhanced simulated annealing (ESA)* [4]: ESA enhances the simulated annealing algorithm by using large steps at high temperature and small steps at low temperature.
3) *Particle swarm optimization (PSO)* [17]: PSO is a new evolutionary computation technique and it exploits the insect swarm behavior.
4) *Evolutionary optimization (EO)* [17]: EO uses a mutation operator and a selection scheme to evolve a population.

TABLE IV
PROBLEM DIMENSIONS ADOPTED IN THE LITERATURE AND IN THIS STUDY

Test function	Problem dimension adopted in the literature	Problem dimension adopted in this study
f_1	30 [5]	30
f_2	30 [5, 17, 18]	30
f_3	30 [5, 18]	30
f_4	30 [5, 17, 18]	30
f_5	30 [5]	30
f_6	30 [5]	30
f_7	10 [14-15]	100
f_8	30 [16]	100
f_9	2, 10 [3]	100
f_{10}	30 [17, 18]; 100 [4]	100
f_{11}	30 [17, 18]	30
f_{12}	30 [18]	30
f_{13}	30 [18]	30
f_{14}	30 [18]	30
f_{15}	30 [18]	30

5) *Conventional evolutionary programming (CEP)* with one of the following mutation operators [18]:
   a) *Guassian mutation operator (CEP/GMO)*: CEP/GMO adopts the Gaussian mutation operator for fast local convergence on convex functions.
   b) *Cauchy mutation operator (CEP/CMO)*: CEP/CMO adopts the Cauchy mutation operator for effective escape from the local optima.
   c) *Mean mutation operator (CEP/MMO)*: In CEP/MMO, the mutation operator is a linear combination of Guassian mutation and Cauchy mutation.
   d) *Adaptive mean mutation operator (CEP/AMMO)*: In CEP/AMMO, the mutation operator is an adaptive version of the mean mutation operator.

Each of the above algorithms was executed to solve some of the test functions f_1–f_{15} , and the results were reported in [4], [5], [17], and [18]. We will use these existing results for a direct comparison in Section IV-E.

C. Control Experiment

To identify any improvement due to orthogonal design and quantization, we design and carry out the following control experiment. We execute a conventional genetic algorithm (CGA) to solve the test functions, where CGA is the same as OGA/Q, except for the following differences. CGA does not apply orthogonal design and quantization. It samples the initial population randomly, and its crossover operator samples the potential offspring randomly.

D. Parameter Values

We adopt the following parameter values or scheme.

TABLE V
PERFORMANCE OF OGA/Q

Test function	Mean number of function evaluations	Mean function value	Standard deviation of function value	Globally minimal function value
f_1	302,166	−12569.4537	6.447×10^{-4}	−12569.5
f_2	224,710	0	0	0
f_3	112,421	4.440×10^{-16}	3.989×10^{-17}	0
f_4	134,000	0	0	0
f_5	134,556	6.019×10^{-6}	1.159×10^{-6}	0
f_6	134,143	1.869×10^{-4}	2.615×10^{-5}	0
f_7	302,773	−92.83	2.626×10^{-2}	−99.2784
f_8	190,031	4.672×10^{-7}	1.293×10^{-7}	0
f_9	245,930	−78.3000296	6.288×10^{-3}	−78.33236
f_{10}	167,863	7.520×10^{-1}	1.140×10^{-1}	0
f_{11}	112,559	0	0	0
f_{12}	112,652	6.301×10^{-3}	4.069×10^{-4}	0
f_{13}	112,612	0	0	0
f_{14}	112,576	0	0	0
f_{15}	112,893	0	0	0

- *Parameters for generating initial population:* The feasible solution space is divided into S subspaces where

$$S = \begin{cases} 10, & \text{if } \max\limits_{1 \le i \le N} (u_i - l_i) \le 100 \\ 20, & \text{if } \max\limits_{1 \le i \le N} (u_i - l_i) > 100. \end{cases}$$

  In other words, a large feasible solution space is divided into more subspaces. The number of quantization levels Q_1 is $N - 1$. Consequently, J_1 is found to be 2 based on (5), so that the orthogonal array $L_{(N-1)^2}((N-1)^N)$ is applied to generate $(N-1)^2$ points in each subspace.
- *Population size:* Since the problem dimensions are high, we choose a moderate population size $G = 200$.
- *Crossover parameters:* We choose $p_c = 0.10$. The number of quantization levels Q_2 is 3, and there are $F = 4$ factors. Consequently, J_2 can be found to be 2, so that each crossover operator applies the orthogonal array $L_{3^2}(3^4)$ to produce nine potential offspring. With these parameter values, the algorithm generates a reasonable number of potential offspring in each generation.
- *Mutation probability:* We choose $p_m = 0.02$, which is significantly smaller than p_c.
- *Stopping criterion:* When the smallest cost of the chromosomes cannot be further reduced in successive 50 generations after 1000 generations, the execution of the algorithm is stopped.

Remark: Recall that the dimension of test functions f_7–f_{10} is 100. When we use the above parameter values to generate an initial population for these test functions, the orthogonal array has 9801 rows, and hence it samples 9801 points. In 100 dimensions, this number is relatively small compared with the size of the feasible solution space and the number of local minima. For

TABLE VI
COMPARISON BETWEEN OGA/Q AND CGA WHERE CGA IS EXECUTED TO SOLVE THE TEST FUNCTIONS IN THE CONTROL EQUIPMENT

Test Function	Mean number of function evaluations		Mean function value (standard deviation)		Globally minimal function value
	OGA/Q	CGA	OGA/Q	CGA	
f_1	302,166	458,653	−12569.4537 (6.447×10^{-4})	−8444.7583 (65.7326)	−12569.5
f_2	224,710	335,993	0 (0)	22.967 (0.7800)	0
f_3	112,421	336,481	4.440×10^{-16} (3.989×10^{-17})	2.697 (5.668×10^{-2})	0
f_4	134,000	346,971	0 (0)	1.258 (1.657×10^{-2})	0
f_5	134,556	346,800	6.019×10^{-6} (1.159×10^{-6})	3.739×10^{-1} (7.730×10^{-3})	0
f_6	134,143	348,356	1.869×10^{-4} (2.615×10^{-5})	2.978 (7.210×10^{-2})	0
f_7	302,773	338,417	−92.83 (2.626×10^{-2})	−83.2739 (1.638×10^{-1})	−99.2784
f_8	190,031	212,601	4.672×10^{-7} (1.293×10^{-7})	82064.7112 (2177.3876)	0
f_9	245,930	268,286	−78.3000296 (6.288×10^{-3})	−59.052839 (6.360×10^{-2})	−78.33236
f_{10}	167,863	1,651,448	7.520×10^{-1} (1.140×10^{-1})	150.79348 (9.299×10^{-1})	0
f_{11}	112,559	181,445	0 (0)	4.9655 (11.3614)	0
f_{12}	112,652	193,284	6.301×10^{-3} (4.069×10^{-4})	5.8524 (1.9722)	0
f_{13}	112,612	170,955	0 (0)	7.9315×10^{-1} (5.5943×10^{-1})	0
f_{14}	112,576	203,143	0 (0)	18.8293 (12.6129)	0
f_{15}	112,893	185,373	0 (0)	2.6205 (1.0261)	0

f_7 as an example, if the solution is to be accurate up to two decimal places, there are $(\pi/0.01)^{100} = 5.19 \times 10^{249}$ possible points in the feasible solution space, and there are $100! = 9.33 \times 10^{157}$ local minima. In the literature, only a few researchers tackled the test functions with more than 30 dimensions (e.g., see Table IV) and consequently, their algorithms sample a smaller number of points for the initial population.

E. Results and Comparison

We performed 50 independent runs for each algorithm on each test function and recorded: 1) the mean number of function evaluations, 2) the mean function value (i.e., the mean of the function values found in the 50 runs), and 3) the standard deviation of the function values.

Table V shows the performance of OGA/Q. We see that the mean function values are equal or close to the optimal ones, and the standard deviations of the function values are relatively small. Consider f_1 as an example. The mean function value is -12569.4537, which is close to the global minimum -12569.5, while the standard deviation of function value is only 6.447×10^{-4}. These results indicate that OGA/Q can find optimal or close-to-optimal solutions, and its solution quality is quite stable.

Table VI shows the results of the control experiment. Recall that CGA is the same as OGA/Q, except that it uses random

TABLE VII
COMPARISON BETWEEN OGA/Q AND FES WHERE THE RESULTS FOR FES ARE OBTAINED FROM [5]

Test Function	Mean number of function evaluations		Mean function value (standard deviation)		Globally minimal function value
	OGA/Q	FES [5]	OGA/Q	FES [5]	
f_1	302,166	900,030	-12569.4537 (6.447×10^{-4})	-12556.4 (32.53)	-12569.5
f_2	224,710	500,030	0 (0)	0.16 (0.33)	0
f_3	112,421	150,030	4.440×10^{-16} (3.989×10^{-17})	1.2×10^{-2} (1.8×10^{-3})	0
f_4	134,000	200,030	0 (0)	3.7×10^{-2} (5.0×10^{-2})	0
f_5	134,556	150,030	6.019×10^{-6} (1.159×10^{-6})	2.8×10^{-6} (8.1×10^{-7})	0
f_6	134,143	150,030	1.869×10^{-4} (2.615×10^{-5})	4.7×10^{-5} (1.5×10^{-5})	0

TABLE VIII
COMPARISON BETWEEN OGA/Q AND ESA WHERE THE RESULTS FOR ESA ARE OBTAINED FROM [4]

Test Function	Mean number of function evaluations		Mean function value (standard deviation)		Globally minimal function value
	OGA/Q	ESA [4]	OGA/Q	ESA [4]	
f_{10}	167,863	188,227	7.520×10^{-1} (1.140×10^{-1})	17.10 (NA)	0

sampling instead of orthogonal sampling. We see that CGA requires more function evaluations than OGA/Q, and hence it has a larger time complexity. However, CGA gives larger mean function values than OGA/Q, and hence its mean solution quality is poorer. In addition, CGA gives larger standard deviations of function values than OGA/Q, and hence its solution quality is less stable. These results indicate that orthogonal design and quantization can effectively improve the genetic algorithm.

In Tables VII–XIII, we compare the performance of OGA/Q with the five existing algorithms. Since each of these existing algorithms was executed to solve *some* of the test functions f_1–f_{15} in [4], [5], [17], and [18], these tables already include all of the available results for comparison.

Table VII compares OGA/Q with FES. For f_1–f_4, OGA/Q can give smaller mean function values using smaller numbers of function evaluations. For f_5–f_6, OGA/Q can find close-to-optimal solutions, but FES can find closer-to-optimal solutions using more function evaluations. For f_5 as an example, OGA/Q gives a mean function value of 6.019×10^{-6}, which is already very close to the global minimum 0, but FES gives a mean function value of 2.8×10^{-6} using more function evaluations.

Tables VIII–XIII compare OGA/Q with ESA, PSO, EO, and the four versions of CEP. For some test functions (e.g., f_{10} with 100 dimensions),OGA/Q can give significantly better and closer-to-optimal solutions; for the other test functions, both OGA/Q and the existing algorithms can give close-to-optimal

TABLE IX
COMPARISON BETWEEN OGA/Q AND PSO AND EO WHERE THE RESULTS FOR PSO AND EO ARE OBTAINED FROM [17]; IN THIS EXPERIMENT THE DIMENSION OF f_{10} IS 30, THE SAME AS THE DIMENSION ADOPTED IN [17]

Test Function	Mean number of function evaluations			Mean function value (standard deviation)			Globally minimal function value
	OGA/Q	PSO [17]	EO [17]	OGA/Q	PSO [17]	EO [17]	
f_2	224,710	250,000	250,000	0 (0)	47.1354 (1.8782)	46.4689 (2.4545)	0
f_4	134,000	250,000	250,000	0 (0)	0.4498 (0.0566)	0.4033 (0.0436)	0
f_{10}	144,872	250,000	250,000	6.891×10^{-3} (1.949×10^{-3})	1911.598 (374.2935)	1610.359 (293.5783)	0
f_{11}	112,559	250,000	250,000	0 (0)	11.175 (1.3208)	9.8808 (0.9444)	0

TABLE X
COMPARISON BETWEEN OGA/Q AND CEP/GMO WHERE THE RESULTS FOR CEP/GMO ARE OBTAINED FROM [18]; IN THIS EXPERIMENT THE DIMENSION OF f_{10} IS 30, THE SAME AS THE DIMENSION ADOPTED IN [18]

Test Function	Mean number of function evaluations			Mean function value (standard deviation)			Globally minimal function value
	OGA/Q	CEP/GMO [18]		OGA/Q	CEP/GMO [18]		
		$b=0$	$b=10^{-4}$		$b=0$	$b=10^{-4}$	
f_2	224,710	250,000	250,000	0 (0)	113.72 (NA)	120.00 (NA)	0
f_3	112,421	150,000	150,000	4.440×10^{-16} (3.989×10^{-17})	10.47 (NA)	9.10 (NA)	0
f_4	134,000	250,000	250,000	0 (0)	6.05 (NA)	2.52×10^{-7} (NA)	0
f_{10}	144,872	250,000	250,000	6.891×10^{-3} (1.949×10^{-3})	2.33×10^{4} (NA)	86.70 (NA)	0
f_{11}	112,559	150,000	150,000	0 (0)	83.53 (NA)	3.09×10^{-7} (NA)	0
f_{12}	112,652	250,000	250,000	6.301×10^{-3} (4.069×10^{-4})	44.15 (NA)	12.20 (NA)	0
f_{13}	112,612	250,000	250,000	0 (0)	9.74×10^{3} (NA)	1.99×10^{-3} (NA)	0
f_{14}	112,576	250,000	250,000	0 (0)	2920.63 (NA)	17.60 (NA)	0
f_{15}	112,893	250,000	250,000	0 (0)	5.89 (NA)	5.18 (NA)	0

solutions. These results indicate that OGA/Q can, in general, give better mean solution quality. In addition, OGA/Q requires smaller mean numbers of function evaluations than the existing algorithms, and hence it has a smaller time complexity. Furthermore, OGA/Q gives smaller standard deviation of function values than PSO and EO (the standard deviation given by ESA and CEP are not available in [4] and [18]), and hence it has a more stable solution quality.

V. CONCLUSION

We designed the orthogonal genetic algorithm with quantization (OGA/Q) for global numerical optimization with continuous variables. Our objective was to apply the experimental de-

TABLE XI
COMPARISON BETWEEN OGA/Q AND CEP/CMO WHERE THE RESULTS FOR CEP/CMO ARE OBTAINED FROM [18]; b IS A CONTROL PARAMETER IN CEP; IN THIS EXPERIMENT THE DIMENSION OF f_{10} IS 30, THE SAME AS THE DIMENSION ADOPTED IN [18]

Test Function	Mean number of function evaluations			Mean function value (standard deviation)			Globally minimal function value
	OGA/Q	CEP/CMO [18]		OGA/Q	CEP/CMO [18]		
		$b=0$	$b=10^{-4}$		$b=0$	$b=10^{-4}$	
f_2	224,710	250,000	250,000	0 (0)	55.82 (NA)	4.73 (NA)	0
f_3	112,421	150,000	150,000	4.440×10^{-16} (3.989×10^{-17})	4.52 (NA)	1.30×10^{-3} (NA)	0
f_4	134,000	250,000	250,000	0 (0)	7.08 (NA)	2.20×10^{-6} (NA)	0
f_{10}	144,872	250,000	250,000	6.891×10^{-3} (1.949×10^{-3})	4.37×10^{3} (NA)	114.00 (NA)	0
f_{11}	112,559	150,000	150,000	0 (0)	104.74 (NA)	3.07×10^{-6} (NA)	0
f_{12}	112,652	250,000	250,000	6.301×10^{-3} (4.069×10^{-4})	42.01 (NA)	9.42 (NA)	0
f_{13}	112,612	250,000	250,000	0 (0)	4.23×10^{3} (NA)	5.87×10^{-3} (NA)	0
f_{14}	112,576	250,000	250,000	0 (0)	2740.70 (NA)	5.78 (NA)	0
f_{15}	112,893	250,000	250,000	0 (0)	5.89 (NA)	6.60×10^{-1} (NA)	0

TABLE XIII
COMPARISON BETWEEN OGA/Q AND CEP/AMMO WHERE THE RESULTS FOR CEP/AMMO ARE OBTAINED FROM [18]; b IS A CONTROL PARAMETER IN CEP/AMMO; IN THIS EXPERIMENT THE DIMENSION OF f_{10} IS 30, THE SAME AS THE DIMENSION ADOPTED IN [18]

Test Function	Mean number of function evaluations			Mean function value (standard deviation)			Globally minimal function value
	OGA/Q	CEP/AMMO [18]		OGA/Q	CEP/AMMO [18]		
		$b=0$	$b=10^{-4}$		$b=0$	$b=10^{-4}$	
f_2	224,710	250,000	250,000	0 (0)	52.59 (NA)	11.90 (NA)	0
f_3	112,421	150,000	150,000	4.440×10^{-16} (3.989×10^{-17})	3.41 (NA)	9.43×10^{-4} (NA)	0
f_4	134,000	250,000	250,000	0 (0)	1.85 (NA)	1.02×10^{-6} (NA)	0
f_{10}	144,872	250,000	250,000	6.891×10^{-3} (1.949×10^{-3})	1.45×10^{3} (NA)	1.44×10^{2} (NA)	0
f_{11}	112,559	150,000	150,000	0 (0)	38.22 (NA)	1.61×10^{-6} (NA)	0
f_{12}	112,652	250,000	250,000	6.301×10^{-3} (4.069×10^{-4})	14.86 (NA)	9.64 (NA)	0
f_{13}	112,612	250,000	250,000	0 (0)	81.97 (NA)	3.99×10^{-3} (NA)	0
f_{14}	112,576	250,000	250,000	0 (0)	1545.01 (NA)	6.12×10^{-1} (NA)	0
f_{15}	112,893	250,000	250,000	0 (0)	3.14 (NA)	3.23×10^{-1} (NA)	0

TABLE XII
COMPARISON BETWEEN OGA/Q AND CEP/MMO WHERE THE RESULTS FOR CEP/MMO ARE OBTAINED FROM [18]; b IS A CONTROL PARAMETER IN CEP/MMO; IN THIS EXPERIMENT, THE DIMENSION OF f_{10} IS 30, THE SAME AS THE DIMENSION ADOPTED IN [18]

Test Function	Mean number of function evaluations			Mean function value (standard deviation)			Globally minimal function value
	OGA/Q	CEP/MMO [18]		OGA/Q	CEP/MMO [18]		
		$b=0$	$b=10^{-4}$		$b=0$	$b=10^{-4}$	
f_2	224,710	250,000	250,000	0 (0)	46.96 (NA)	9.52 (NA)	0
f_3	112,421	150,000	150,000	4.440×10^{-16} (3.989×10^{-17})	3.75 (NA)	7.49×10^{-4} (NA)	0
f_4	134,000	250,000	250,000	0 (0)	3.84 (NA)	6.99×10^{-7} (NA)	0
f_{10}	144,872	250,000	250,000	6.891×10^{-3} (1.949×10^{-3})	2.87×10^{3} (NA)	63.8 (NA)	0
f_{11}	112,559	150,000	150,000	0 (0)	41.12 (NA)	9.81×10^{-7} (NA)	0
f_{12}	112,652	250,000	250,000	6.301×10^{-3} (4.069×10^{-4})	33.98 (NA)	9.53 (NA)	0
f_{13}	112,612	250,000	250,000	0 (0)	5.55×10^{3} (NA)	3.23×10^{-3} (NA)	0
f_{14}	112,576	250,000	250,000	0 (0)	2812.26 (NA)	11.80 (NA)	0
f_{15}	112,893	250,000	250,000	0 (0)	6.31 (NA)	1.88 (NA)	0

sign methods to enhance the genetic algorithm, so that it could be more robust and statistically sound. In particular, we proposed a quantization technique to complement the orthogonal design, and we applied the resulting methodology to design a new method for generating a good initial population and a new crossover operator. We executed OGA/Q to solve 15 benchmark problems. The dimensions of these problems are 30 or 100, and some of them have numerous local minima. The results show that OGA/Q can find optimal or close-to-optimal solutions, and it is more competitive than five recent algorithms on the problems studied.

ACKNOWLEDGMENT

The authors sincerely thank the reviewers for their constructive comments, and Dr. D. B. Fogel for his suggestions on improving the presentation.

REFERENCES

[1] W. W. Hager, W. Hearn, and P. M. Pardalos, Eds., *Large Scale Optimization: State of the Art*. Norwell, MA: Kluwer, 1994.

[2] C. A. Floudas and P. M. Pardalos, Eds., *State of the Art in Global Optimization: Computational Methods and Applications*. Norwell, MA: Kluwer, 1996.

[3] M. A. Stybinski and T. S. Tang, "Experiments in nonconvex optimization: Stochastic approximation and function smoothing and simulated annealing," *Neural Networks*, vol. 3, pp. 467–483, 1990.

[4] P. Siarry, G. Berthiau, F. Durbin, and J. Haussy, "Enhanced simulated annealing for globally minimizing functions of many-continuous variables," *ACM Trans. Math. Software*, vol. 23, pp. 209–228, June 1997.

[5] X. Yao and Y. Liu, "Fast evolution strategies," in *Evolutionary Programming VI*, P. J. Angeline, R. Reynolds, J. McDonnell, and R. Eberhart, Eds. Berlin, Germany: Springer-Verlag, 1997, pp. 151–161. (Available at ftp://www.cs.adfa.edu.au/pub/xin/yao_liu_ep97.ps.gz).

[6] D. E. Goldberg, *Genetic Algorithm in Search, Optimization and Machine Learning*. Reading, MA: Addison-Wesley, 1989.

[7] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, 3rd ed. Berlin, Germany: Springer-Verlag, 1996.

[8] Y. W. Leung and Q. Zhang, "Evolutionary algorithms + experimental design methods: A hybrid approach for hard optimization and search problems," Res. Grant Proposal, Hong Kong Baptist Univ., (A short version is available at http://www.comp.hkbu.edu.hk/~ywleung/prop/EA_EDM.doc), 1997.

[9] D. C. Montgomery, *Design and Analysis of Experiments*, 3rd ed. New York: Wiley, 1991.

[10] C. R. Hicks, *Fundamental Concepts in the Design of Experiments*, 4th ed. TX: Saunders College Publishing, 1993.

[11] Q. Zhang and Y. W. Leung, "An orthogonal genetic algorithm for multimedia multicast routing," *IEEE Trans. Evol. Comput.*, vol. 3, pp. 53–62, Apr. 1999.

[12] Q. Wu, "On the optimality of orthogonal experimental design," *Acta Math. Appl. Sinica*, vol. 1, no. 4, pp. 283–299, 1978.

[13] Math. Stat. Res. Group, Chinese Acad. Sci., *Orthogonal Design* (in Chinese). Beijing: People Education Pub., 1975.

[14] J. Yen and B. Lee, "A simplex genetic algorithm hybrid," in *Proc. 1997 IEEE Int. Conf. Evolutionary Computation*, IN, Apr. 1997, pp. 175–180.

[15] J. Renders and H. Bersini, "Hybridizing genetic algorithms with hill-climbing methods for global optimization: Two possible ways," in *Proc. 1st IEEE Conf. Evolutionary Computation*, Orlando, FL, June 1994, pp. 312–317.

[16] H. P. Schwefel, *Evolution and Optimum Seeking*. New York: Wiley, 1995.

[17] P. J. Angeline, "Evolutionary optimization versus particle swarm optimization: Philosophy and performance differences," in *Proc. Evol. Prog. VII*, V. W. Porto, N. Saravanan, D. Waagen, and A. E. Eiben, Eds. Berlin, Germany: Springer-Verlag, 1998, pp. 601–610.

[18] K. Chellapilla, "Combining mutation operators in evolutionary programming," *IEEE Trans. Evol. Comput.*, vol. 2, pp. 91–96, Sept 1998.