Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2012, Article ID 836597, 27 pages
doi:10.1155/2012/836597

Research Article

Improved Quantum-Inspired Evolutionary Algorithm for Engineering Design Optimization

Jinn-Tsong Tsai,[1] Jyh-Horng Chou,[2,3] and Wen-Hsien Ho[4]

[1] *Department of Computer Science, National Pingtung University of Education, 4-18 Min-Sheng Road, Pingtung 900, Taiwan*
[2] *Institute of System Information and Control, National Kaohsiung First University of Science and Technology, 1 University Road, Yenchao, Kaohsiung 824, Taiwan*
[3] *Department of Electrical Engineering, National Kaohsiung University of Applied Sciences, 415 Chien-Kung Road, Kaohsiung 807, Taiwan*
[4] *Department of Healthcare Administration and Medical Informatics, Kaohsiung Medical University, 100 Shi-Chuan 1st Road, Kaohsiung 807, Taiwan*

Correspondence should be addressed to Wen-Hsien Ho, whho@kmu.edu.tw

Received 31 August 2012; Revised 26 October 2012; Accepted 31 October 2012

Academic Editor: Jung-Fa Tsai

An improved quantum-inspired evolutionary algorithm is proposed for solving mixed discrete-continuous nonlinear problems in engineering design. The proposed Latin square quantum-inspired evolutionary algorithm (LSQEA) combines Latin squares and quantum-inspired genetic algorithm (QGA). The novel contribution of the proposed LSQEA is the use of a QGA to explore the optimal feasible region in macrospace and the use of a systematic reasoning mechanism of the Latin square to exploit the better solution in microspace. By combining the advantages of exploration and exploitation, the LSQEA provides higher computational efficiency and robustness compared to QGA and real-coded GA when solving global numerical optimization problems with continuous variables. Additionally, the proposed LSQEA approach effectively solves mixed discrete-continuous nonlinear design optimization problems in which the design variables are integers, discrete values, and continuous values. The computational experiments show that the proposed LSQEA approach obtains better results compared to existing methods reported in the literature.

1. Introduction

Solving engineering design optimization problems usually requires consideration of many different types of design variables and many constraints. Practical problems in engineering design often involve a mix of integers, discrete variables, and continuous variables. These constraints are often problematic during the engineering design optimization process.

Since the 1960s, researchers have attempted to solve this problem, which is known as the mixed discrete nonlinear programming (MDNLP) problem. One of the most effective solutions reported so far is a nonlinear branch and bound method (BBM) for solving nonlinear and discrete programming in mechanical design optimization [1, 2]. In BBM, however, subproblems result from portioning the feasible domain to obtain solutions by ignoring discrete conditions, and the number of times the problem needs to be resolved increases exponentially with the number of variables [3]. The better method, such as the sequential linear programming (SLP), was developed by Bremicker et al. [4] and by Loh and Papalambros [5] to solve general MDNLP problems. The linearized discrete problem is solved by the simplex method to obtain information at each node of the tree. Their SLP approach is compared with the pure BBM, where the sequential quadratic programming is used to solve the nonlinear problem to obtain information at each node. The study shows the SLP method to be superior to the pure BBM. Other approaches to solving MDNLP problems include the penalty function approach [6–8] and the Lagrangian relaxation approach [9]. The penalty function approach to treat the requirement of discreteness is to define additional constraints and construct a penalty function for them. This term imposes penalty for deviations from the discrete values. The difficulties with a penalty approach are the introduction of additional local minima and repeated minimizations by adjusting the penalty parameters [3]. The Lagrangian relaxation method is similar to the penalty function method. The main difference is that the additional terms due to discrete variables are added to a Lagrangian function instead of a penalty function. The Lagrangian relaxation approach does not guarantee finding a global solution, even if it is a convex problem before discrete variables are introduced. It is observed that some of the methods for discrete variable optimization use the structure of the problem to speed up the search for the discrete solution. These methods are not suitable for implementation into general purpose applications. The BBM is the most general methods; however, it is time consuming. In recent years, the focus has shifted to applications of soft-computing optimization techniques that naturally use mixed-discrete and continuous variables for solving practical engineering problems. These approaches include genetic algorithms (GAs) [10–18], simulated annealing [19], differential evolution [20, 21], and evolutionary programming approach [22]. The major challenge when solving MDNLP problems is that numerous local optima can result in the methods becoming trapped in the local optima of the objective functions [12]. Therefore, an efficient and robust algorithm is needed to solve mixed discrete-continuous nonlinear design optimization problems in the engineering design field.

In the past decade, the emerging field of quantum-inspired computing has motivated intensive studies of algorithms such as the Shor factorizing algorithm [23] and the Grover quantum search algorithm [24, 25]. By applying quantum mechanical principles such as quantum-bit representation and superposition of states, quantum-inspired computing can simultaneously process huge numbers of quantum states simultaneously and in parallel. To introduce a strong parallelism in the evolutionary algorithm, Han et al. [26] and Han and Kim [27–29] proposed the quantum-inspired genetic algorithm (QGA). For solving combinatorial optimization problems, QGA has proven superior to conventional GAs. Malossini et al. [30] showed that, by taking advantage of quantum phenomena, QGA improves the speed and efficiency of genetic procedures. Quantum-inspired evolutionary algorithms have also been used to solve optimization problems such as partition function calculation [31], nonlinear blind source separation [32], filter design [33], numerical optimization [34–36], hyperspectral anomaly detection [37], multiple sequence alignment [38], thermal process identification [39], and multiobjective optimization [40]. However, the performance of the simple QGA

is often unsatisfactory, and it is easily trapped in the local optima, which results in premature convergence. That is, the quantum-inspired bit (Q-bit) search with quantum mechanism must be well coordinated with the genetic search with evolution mechanism, and the exploration and exploitation behaviors must also be well balanced [35]. Therefore, a big challenge is to improve QGA capability of exploration and exploitation and develop an efficient and robust algorithm.

The efficient and robust Latin square quantum-inspired evolutionary algorithm (LSQEA) proposed in this study solves global numerical optimization problems with continuous variables and mixed discrete-continuous nonlinear design optimization problems. The LSQEA approach integrates Latin squares [41–43] and QGA (i.e., quantum-inspired individual and mechanism with GAs). The concept of the use of QGA came from the works of Han et al. [26] and Han and Kim [27–29], while the development steps were implemented by authors and shown in Section 3 . The role of the Latin square is to generate better individuals by implementing the Latin square-based recombination since the systematic reasoning ability of Latin square of the Taguchi method is, in due course, incorporated into the recombination operation to select better Q-bits. This role is important for improving the efficiency of the crossover operation in generating representative individuals and better-fit trial individuals. The Latin square is applied to recombine the better Q-bits so that potential individuals in microspace can be exploited. The QGA is used to explore the optimal feasible region in macrospace. Therefore, the LSQEA approach is highly robust and achieves quick convergence.

The paper is organized as follows. Section 2 gives the problem statements. The LSQEA for solving the mixed discrete-continuous nonlinear design optimization problems is described in Section 3. In Section 4, the proposed LSQEA approach is compared with QGA and real-coded GA (RGA) [44–47] in terms of performance in solving global numerical optimization problems with continuous variables. The LSQEA approach is then used to solve mixed discrete-continuous nonlinear design problems encountered in the engineering design field, and the results obtained by LSQEA are compared with those obtained by existing methods reported in the literature. Finally, Section 5 concludes the study.

2. Problem Statements

This section states the considered problems, which include a global numerical optimization problem with continuous variables and a mixed discrete-continuous nonlinear programming problem.

The following global numerical optimization problem with continuous variables is considered:

$$\begin{aligned} &\text{minimize} \quad f(X) \\ &\text{subject to} \quad X^L \le X \le X^U, \\ &\qquad\qquad g_j(X) \le 0 \quad (j = 1, 2, \ldots, t), \end{aligned} \tag{2.1}$$

where $X = [x_1, x_2, \ldots, x_i, \ldots, x_n]$ is a continuous variable vector, $f(X)$ is an objective function, $X^L = [x_1^L, \ldots, x_i^L, \ldots, x_n^L]$, and $X^U = [x_1^U, \ldots, x_i^U, \ldots, x_n^U]$ define the feasible solution vector spaces. The domain of x_i is denoted by $[x_i^L, x_i^U]$, and the feasible solution space is defined by $[X^L, X^U]$. For this problem, $g_j(X)$ and $j = 1, 2, \ldots, t$ are the constraint functions. Although many design problems can be cast as the above optimization problem,

efficiently obtaining optimal solution is difficult because the problem involves designs that are high-dimensional, nondifferentiable, and multimodal [35].

The mixed discrete-continuous nonlinear programming problem is expressed as follows [12]:

$$\begin{aligned} &\text{minimize} \quad f(X) \\ &\text{subject to} \quad g_j(X) \le 0 \quad (j = 1, 2, \ldots, t), \\ &\qquad x_i^L \le x_i \le x_i^U \quad (i = 1, 2, \ldots, n), \\ &\qquad X = [X^d, X^c]^T, \\ &\qquad X^d = \left[x_1, x_2, \ldots, x_{nq}, x_{nq+1}, \ldots, x_{nd}\right]^T, \\ &\qquad X^c = [x_{nd+1}, x_{nd+2}, \ldots, x_n]^T, \end{aligned} \tag{2.2}$$

where $x_1, x_2, \ldots, x_{nq}$ are nonnegative discrete variables with permissible values equally spaced; $x_{nq+1}, \ldots, x_{nd}$ are nonnegative discrete variables with permissible values unequally spaced; $x_{nd+1}, x_{nd+2}, \ldots, x_n$ are nonnegative continuous variables. Practical optimization problems encountered in the engineering design field often have many constraints and require consideration of different types of design variables as shown in (2.2). Again, because these problems involve high-dimensional, nondifferentiable, and multimodal properties, an effective algorithm is needed to solve them optimally and efficiently. According to the above statements, the problem in (2.2) is a special case of the problem in (2.1).

For the mixed discrete-continuous design problem in (2.2), the discrete variables with equal spacing (i.e., individuals with arithmetical progression) are $x_i = x_i^L + (N_i - 1)\Delta x_i$, where $i = 1, 2, \ldots, nq$; x_i^L is the lower bound of x_i; N_i is the natural number corresponding to x_i; Δx_i is the discrete increment of the ith discrete variable; nq is the number of discrete variables with equal spacing. Let $x_i = e_{M_i,1}$ for the discrete variables with unequal spacing, where $nq < i \le nd$, M_i denotes the natural number corresponding to x_i, M_i is generated from $[1, w_i]$, $e_{M_i,1}$ denotes the M_ith element of the $w_i \times 1$ vector E_i, E_i represents the $w_i \times 1$ vector of values of discrete variables with unequal spacing, and w_i is the maximum permissible number of discrete values for the ith discrete variable with unequal spacing.

3. The LSQEA Approach for Solving Mixed Discrete-Continuous Design Problems

This section describes the details of the LSQEA approach for solving mixed discrete-continuous nonlinear programming problems.

3.1. Q-Bit Representation

In quantum-inspired computing, the smallest unit of information stored in a two-state quantum computer is called a quantum bit. It may be in a "0" state, a "1" state, or any superposition of the two. The state of a quantum bit can be represented as

$$|\Psi\rangle = \alpha|0\rangle + \beta|1\rangle, \tag{3.1}$$

where α and β are complex numbers that describe the probability amplitudes of the corresponding states. The $|\alpha|^2$ and $|\beta|^2$ are probabilities of the quantum bit being in the "0" state and "1" state, respectively, such that $|\alpha|^2 + |\beta|^2 = 1$.

The use of a Q-bit to represent an individual in this study was inspired by quantum computing concepts. The advantage of the representation is the capability to use linear superposition method to generate any possible solution. A Q-bit individual can be represented by a string of n Q-bits such as

$$\left[\begin{array}{c|c|c} \alpha_1 & \alpha_2 \cdots & \alpha_n \\ \beta_1 & \beta_2 \cdots & \beta_n \end{array}\right], \tag{3.2}$$

where $|\alpha_i|^2 + |\beta_i|^2 = 1,\ i = 1, 2, \ldots, n$. Since Q-bits represent a linear superposition of states, a Q-bit representation provides better population diversity compared to other representations used in evolutionary computing. For example, for following three Q-bits system with three pairs of amplitudes

$$\left[\begin{array}{c|c|c} \dfrac{1}{\sqrt{2}} & \dfrac{1}{\sqrt{2}} & \dfrac{1}{2} \\ \dfrac{1}{\sqrt{2}} & -\dfrac{1}{\sqrt{2}} & \dfrac{\sqrt{3}}{2} \end{array}\right], \tag{3.3}$$

the states of the system can be represented as

$$\frac{1}{4}|000\rangle + \frac{\sqrt{3}}{4}|001\rangle - \frac{1}{4}|010\ \rangle - \frac{\sqrt{3}}{4}|011\rangle + \frac{1}{4}|100\rangle + \frac{\sqrt{3}}{4}|101\rangle - \frac{1}{4}|110\ \rangle - \frac{\sqrt{3}}{4}|111\rangle. \tag{3.4}$$

The above result means that the probabilities to represent the states $|000\rangle, |001\rangle, |010\rangle, |011\rangle, |100\rangle, |101\rangle, |110\rangle$, and $|111\rangle$ are $1/16, 3/16, 1/16, 3/16,\ 1/16,\ 3/16,\ 1/16$, and $3/16$, respectively. By consequence, the above three Q-bits system contains the information of eight states.

3.2. Initial Population

The initialization procedure produces p_s Q-bit individuals where p_s denotes the population size.

3.3. Crossover Operation

The crossover operators are the one cut-point operator, which randomly determines one cut-point and exchanges the cut-point right parts of the Q-bits of two parents to generate new

offspring. If the ith position is selected as one cut-point, the one cut-point crossover operator is used for Q-bits as shown in (3.5)

$$\begin{bmatrix} \alpha_{1,1} & \cdots & \alpha_{1,i} & \alpha_{1,i+1} & \cdots & \alpha_{1,n} \\ \beta_{1,1} & \cdots & \beta_{1,i} & \beta_{1,i+1} & \cdots & \beta_{1,n} \end{bmatrix} \quad \begin{bmatrix} \alpha_{1,1} & \cdots & \alpha_{1,i} & \alpha_{2,i+1} & \cdots & \alpha_{2,n} \\ \beta_{1,1} & \cdots & \beta_{1,i} & \beta_{2,i+1} & \cdots & \beta_{2,n} \end{bmatrix}$$

$$\Longrightarrow \tag{3.5}$$

$$\begin{bmatrix} \alpha_{2,1} & \cdots & \alpha_{2,i} & \alpha_{2,i+1} & \cdots & \alpha_{2,n} \\ \beta_{2,1} & \cdots & \beta_{2,i} & \beta_{2,i+1} & \cdots & \beta_{2,n} \end{bmatrix} \quad \begin{bmatrix} \alpha_{2,1} & \cdots & \alpha_{2,i} & \alpha_{1,i+1} & \cdots & \alpha_{1,n} \\ \beta_{2,1} & \cdots & \beta_{2,i} & \beta_{1,i+1} & \cdots & \beta_{1,n} \end{bmatrix}.$$

For example, for following four Q-bits system, the 2nd position is selected as one cut-point, the crossover operation is shown below

$$\begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{2} & \frac{\sqrt{3}}{2} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & \frac{\sqrt{3}}{2} & -\frac{1}{2} \end{bmatrix} \quad \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & -\frac{1}{2} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & \frac{\sqrt{3}}{2} & -\frac{1}{\sqrt{2}} \end{bmatrix}$$

$$\Longrightarrow \tag{3.6}$$

$$\begin{bmatrix} \frac{\sqrt{3}}{2} & -\frac{1}{\sqrt{2}} & -\frac{1}{2} & \frac{1}{\sqrt{2}} \\ -\frac{1}{2} & \frac{1}{\sqrt{2}} & \frac{\sqrt{3}}{2} & -\frac{1}{\sqrt{2}} \end{bmatrix} \quad \begin{bmatrix} \frac{\sqrt{3}}{2} & -\frac{1}{\sqrt{2}} & \frac{1}{2} & \frac{\sqrt{3}}{2} \\ -\frac{1}{2} & \frac{1}{\sqrt{2}} & \frac{\sqrt{3}}{2} & -\frac{1}{2} \end{bmatrix}.$$

3.4. Mutation Operation

Mutation of Q-bits is performed by randomly determining one position (e.g., position i) and then exchanging the corresponding α_i and β_i

$$\begin{bmatrix} \alpha_1 & \cdots & \alpha_i & \cdots & \alpha_n \\ \beta_1 & \cdots & \beta_i & \cdots & \beta_n \end{bmatrix} \Longrightarrow \begin{bmatrix} \alpha_1 & \cdots & \beta_i & \cdots & \alpha_n \\ \beta_1 & \cdots & \alpha_i & \cdots & \beta_n \end{bmatrix}. \tag{3.7}$$

For example, for following four Q-bits system, the 2nd position is selected for mutation, the mutation operation is shown below.

$$\begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{2} & \frac{\sqrt{3}}{2} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & \frac{\sqrt{3}}{2} & -\frac{1}{2} \end{bmatrix} \Longrightarrow \begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & -\frac{1}{2} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{\sqrt{3}}{2} & -\frac{1}{\sqrt{2}} \end{bmatrix}. \tag{3.8}$$

3.5. Q-Bit Rotation Operation

The purpose of a rotation gate $U(\theta)$ is to update a Q-bit individual by rotating the Q-bit toward the direction of the corresponding Q-bit to obtain a better value. The (α_i, β_i) of the ith Q-bit is updated as follows:

$$\begin{bmatrix} \alpha_i' \\ \beta_i' \end{bmatrix} = U(\theta_i) \begin{bmatrix} \alpha_i \\ \beta_i \end{bmatrix} = \begin{bmatrix} \cos\theta_i & -\sin\theta_i \\ \sin\theta_i & \cos\theta_i \end{bmatrix} \begin{bmatrix} \alpha_i \\ \beta_i \end{bmatrix}, \tag{3.9}$$

where θ_i is a rotation angle by $[0, 0.05\pi]$.

For example, if $|\Psi\rangle = 2/\sqrt{5}|0\rangle + 1/\sqrt{5}|1\rangle = 1/\sqrt{5}\begin{pmatrix} 2 \\ 1 \end{pmatrix}$ and $U = 1/\sqrt{2}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$, the result obtained by Q-bit rotation operation is $|\Psi'\rangle = 3/\sqrt{10}|0\rangle + 1/\sqrt{10}|1\rangle$. The probability of $|0$ becomes larger, and the probability of $|1$ becomes smaller.

3.6. Penalty Function

When using evolutionary method to solve a constrained optimization problem, a penalty function is used to relax the constraints by penalizing the unfeasible individuals in the population. This method improves the probability of approaching a feasible region of the search space by navigating through unfeasible regions and by reducing the penalty when a feasible region is approached. To clarify this point, it is important to distinguish between feasible and unfeasible individuals. The unfeasible individuals violate constraints included in the range $[1, R]$ where R is the number of design constraints. The higher this index of violation is, the larger the penalty should be. Given these considerations, the penalty value P is defined as follows:

$$P = w_p \sum_{j=1}^{R} \left| w_L (L_j - y_j) + w_U (y_j - U_j) \right|, \tag{3.10}$$

where y_j is a value computed from the constraint function when the values of design variables are determined; U_j and L_j are the upper and lower bounds, respectively, of the constraint function; w_p is a value distinguishing the feasible from the unfeasible individuals; and w_L and w_U denote the weights. Additionally, if $y_j < L_j$, then $w_L = 1$ and $w_U = 0$; if $y_j > U_j$, then $w_L = 0$ and $w_U = 1$; if $L_j \le y_j \le U_j$, then $w_L = 0$ and $w_U = 0$. Equation (3.10) generally requires that each value calculated for y_j of the constraint function should be limited to its upper and lower bounds. If the value is located within the feasible region, the value is not punished. Otherwise, the value is punished by being multiplied with a large number w_p. The penalty value equals 0 when the optimization process is complete since the values of the design variables no longer violate the design constraints.

3.7. Latin Square

The Latin square experimental design method screens for the important factors that impact product performance. Therefore, it can be used to study a large number of decision variables with a small number of experiments. The design variables (parameters) are called factors, and parameter settings are called levels. The name Latin square originates from Leonhard Euler,

who used Latin characters as symbols. The details regarding the experimental design method can be found in texts by Phadke [41], Montgomery [42], and Park [43]. For an orthogonal array, matrix experiments are conducted by randomly choosing two individuals from the Q-bit population pool. Each factor of one Q-bit individual is designated level 1, and each factor of the other Q-bit individual is designated level 2. The two-level orthogonal array of Latin squares applied here is $L_m(2^{m-1})$. Additionally, each of Z number of design factors has two levels. To establish a two-level orthogonal array of Z factors, let $L_m(2^{m-1})$ represent $m-1$ columns and m individual experiments corresponding to the m rows, where $m = 2^k$, k is a positive integer ($k > 1$) and $Z \leq m-1$. If $Z < m-1$, only the first Z columns are used while the other $m-1-Z$ columns are ignored. For example, if each of six factors has two levels, only six columns are needed to allocate these factors. In this case, $L_8(2^7)$ is sufficient for this purpose because it has seven columns.

The better combinations of decision variables are also determined by integrating the orthogonal array of the Latin square and the signal-to-noise ratio of the Taguchi method. The concept of Taguchi method is to maximize signal-to-noise ratios used as performance measures by using the orthogonal array to run a partial set of experiments. The signal-to-noise ratio (η) refers to the mean-square deviation in the objective function. For cases of the larger-the-better characteristic and the smaller-the-better characteristic, Taguchi defined η, which is expressed in decibels, as $\eta = -10\ \log((1/n)\sum_{i=1}^{n}(1/y_i^2))$ and $\eta = -10\ \log((1/n)\sum_{i=1}^{n}(y_i^2))$, respectively, where $\{y_1, y_2, \ldots, y_n\}$ denotes a set of characteristics. Further details can be found in works by Phadke [41], Montgomery [42], and Park [43].

If only the degree of η in the orthogonal array experiments is being described, the previous equations can be modified as $\eta_i = y_i$ if the objective function is to be maximized (larger-the-better) and as $1/y_i$ if the objective function is to be minimized (smaller-the-better). Let y_i denote the evaluation value of the objective function of experiment i, where $i = 1, 2, \ldots, m$, and m is the number of orthogonal array experiments. The effects of the various factors (variables or individuals) can be defined as follows:

$$E_{fl} = \text{ sum of } \eta_i \text{ for factor } f \text{ at level } l, \tag{3.11}$$

where i is the number of experiments, f is the factor name or number, and l is the level number.

The main objective of the matrix experiments is to choose a new Q-bit individual from the two Q-bit individuals at each locus (factor). At each locus (factor), a Q-bit is chosen if the E_{fl} has the highest value in the experimental region. That is, the objective is to determine the best level for each factor. The best level for a factor is the level that maximizes the value of E_{fl} in the experimental region. For the two-level problem, if $E_{f1} > E_{f2}$, the better level is level 1 for factor $f \in [1, Z]$. Otherwise, level 2 is better. After the best level is determined for each factor, the best levels can be combined to obtain the new individual. Therefore, systematic reasoning ability of the orthogonal array of the Latin square combined with the signal-to-noise ratio of the Taguchi method ensures that the new Q-bit individual has the best or close-to-best evaluation value of the objective function among the 2^Z combinations of factor levels, where 2^Z is the total number of experiments needed for all combinations of factor levels.

For the matrix experiments of an orthogonal array of Latin squares, generation of better individuals requires random selection of two Q-bit individuals at a time from the Q-bit population pool generated by the one cut-point crossover operation. A new individual generated by each matrix experiment is superior to its parents by using the systematic reasoning ability of an orthogonal array of Latin squares and by following the below

algorithm [46]. The two individuals recombine the better Q-bits to be a better-fit individual, so that potential individuals in microspace can be exploited. The detailed steps for each matrix experiment are described as follows.

Algorithm

Step 1. Set $j = 1$. Generate two sets U_1 and U_2, each of which has Z design factors (variables). From the first Z columns of the orthogonal array $L_m(2^{m-1})$, allocate Z design factors, where $m \geq Z + 1$.

Step 2. Designate sets U_1 and U_2 as level 1 and level 2, respectively, by using a uniformly distributed random method to choose two Q-bit individuals from the Q-bit population pool generated by the crossover operation.

Step 3. Assign the level 1 values obtained from U_1 and the level 2 values obtained from U_2 to level cells of the j experiment in the orthogonal array.

Step 4. Calculate the fitness value and the signal-to-noise ratio for the new individual.

Step 5. If $j > m$, then go to Step 6. Otherwise, $j = j + 1$, and repeat Steps 3–5.

Step 6. Calculate the effects of the various factors (E_{f1} and E_{f2}), where $f = 1, 2, \ldots, Z$.

Step 7. The Q-bit of locus f of the new Q-bit individual is obtained from U_1 if $E_{f1} > E_{f2}$. Otherwise, it is obtained from U_2, where $f = 1, 2, \ldots, Z$. Implementing the process for each Q-bit at each locus then obtains the new Q-bit individual.

3.8. Steps of LSQEA

The LSQEA approach is a method of integrating Latin squares and QGA. The Latin square method is performed between the one-cut-point crossover operation and the mutation operation. The penalty function is considered for a constrained problem, as the fitness value is calculated. The steps of the LSQEA approach are described as follows.

Step 1. Set parameters, including population size p_s, crossover rate p_c, mutation rate p_m, and number of generations.

Step 2. Generate an initial Q-bit population, and calculate the fitness values for the population.

Step 3. Perform selection operation by roulette wheel approach [44].

Step 4. Perform the one-cut-point crossover operation for each Q-bit. Select Q-bit individuals for crossover according to crossover rate p_c.

Step 5. Perform matrix experiments for Latin squares method, and use signal-to-noise ratios to generate the better offspring.

Step 6. Repeat Step 5 until the loop number $((1/4) \times p_s \times p_c)$ has been met.

Step 7. Generate the Q-bit population via Latin squares method.

Step 8. Perform the mutation operation in the Q-bit population. Select Q-bits for mutation according to mutation rate p_m.

Step 9. Except for the best individual, select p_s Q-bit individuals for the Q-bit rotation operation.

Step 10. Generate the new Q-bit population.

Step 11. Has the stopping criterion been met? If so, then go to Step 12. Otherwise, repeat Step 3 to Step 11.

Step 12. Display the best individual and fitness value.

4. Design Examples and Comparisons

This section first describes the performance evaluation results for the proposed LSQEA approach. The performance of the LSQEA is then compared with those of the QGA and RGA methods in solving nonlinear programming optimization problems with continuous variables. Finally, the LSQEA approach is used to solve mixed discrete-continuous nonlinear design problems in the engineering design field, and its solutions are compared with those of other methods reported in the literature.

4.1. Solving Nonlinear Programming Optimization Problems with Continuous Variables

For performance evaluation, the proposed LSQEA approach was used to solve the nonlinear programming optimization problems shown in Table 1 [48–50]. The test functions included quadratic, linear, polynomial, and nonlinear forms. The constraints of these functions (f_1, f_2, f_3, and f_4) were linear and nonlinear inequalities, and their dimensions were 13, 8, 7, and 10, respectively. The penalty function of (3.10) was used to handle constrains of linear and nonlinear inequalities for optimization. Therefore, the test functions had sufficient local minima to provide a challenging problem for the purpose of performance evaluation. To identify any performance improvements obtained by application of Latin square and quantum computing-inspired concepts, the QGA and RGA approaches were used to solve the test functions.

Optimizing the main parameters in evolutionary environments continues to be a area of active research in this field. Studies have shown how the performance of a GA can be improved by modifying its main parameters [51, 52]. For example, Chou et al. [53] applied an experimental design method to improve the performance of a GA by optimizing its evolutionary parameters. Therefore, this study adjusted evolutionary parameters by using the same experimental design method applied in Chou et al. [53]. The evolutionary environments used for experimental computation by LSQEA, QGA, and RGA approaches were as follows. For f_1, f_2, f_3, and f_4, the population size p_s was 300, the crossover rate p_c was 0.9, and the mutation rate p_m was 0.1. For f_1, f_2, and f_4, the stopping criterion for all methods and test functions was 540000 function calls. For f_3, however, the stopping criterion was set to only 300000 function calls because it approached the optimal value fastest. Additionally,

Table 1: Test functions.

$f_1(X) = 5x_1 + 5x_2 + 5x_3 + 5x_4 - 5\sum_{i=1}^{4} x_i^2 - \sum_{i=5}^{13} x_i,$
subject to
$2x_1 + 2x_2 + x_{10} + x_{11} \le 10,\ 2x_1 + 2x_3 + x_{10} + x_{12} \le 10,\ -8x_1 + x_{10} \le 0,$
$2x_2 + 2x_3 + x_{11} + x_{12} \le 10,\ -8x_2 + x_{11} \le 0,\ -8x_3 + x_{12} \le 0,$
$-2x_4 - x_5 + x_{10} \le 0,\ -2x_6 - x_7 + x_{11} \le 0,\ -2x_8 - x_9 + x_{12} \le 0,$
$0 \le x_i \le 1,\ i = 1, \ldots, 9,\ 0 \le x_i \le 100,\ i = 10, 11, 12,\ 0 \le x_{13} \le 1.$
$f_2(X) = x_1 + x_2 + x_3,$
subject to
$1 - 0.0025(x_4 + x_6) \ge 0,\ 1 - 0.0025(x_5 + x_7 - x_4) \ge 0,$
$1 - 0.01(x_8 - x_5) \ge 0,\ x_1x_6 - 833.33252x_4 - 100x_1 + 83333.333 \ge 0,$
$x_2x_7 - 1250x_5 - x_2x_4 + 1250x_4 \ge 0,\ x_3x_8 - 1250000 - x_3x_5 + 2500x_5 \ge 0,$
$100 \le x_1 \le 10000,\ 1000 \le x_i \le 10000,\ i = 2, 3,\ 10 \le x_i \le 1000,\ i = 4, \ldots, 8.$
$f_3(X) = (x_1 - 10)^2 + 5(x_2 - 12)^2 + x_3^4 + 3(x_4 - 11)^2 + 10x_5^6 + 7x_6^2 + x_7^4 - 4x_6x_7 - 10x_6 - 8x_7,$
subject to
$127 - 2x_1^2 - 3x_2^4 - x_3 - 4x_4^2 - 5x_5 \ge 0,\ 282 - 7x_1 - 3x_2 - 10x_3^2 - x_4 + x_5 \ge 0,$
$196 - 23x_1 - x_2^2 - 6x_6^2 + 8x_7 \ge 0,\ -4x_1^2 - x_2^2 + 3x_1x_2 - 2x_3^2 - 5x_6 + 11x_7 \ge 0,$
$-10 \le x_i \le 10,\ i = 1, \ldots, 7.$
$f_4(X) = x_1^2 + x_2^2 + x_1x_2 - 14x_1 - 16x_2 + (x_3 - 10)^2 + 4(x_4 - 5)^2 + (x_5 - 3)^2 + 2(x_6 - 1)^2 + 5x_7^2$ $+7(x_8 - 11)^2 + 2(x_9 - 10)^2 + (x_{10} - 7)^2 + 45,$
subject to
$105 - 4x_1 - 5x_2 + 3x_7 - 9x_8 \ge 0,\ -10x_1 + 8x_2 + 17x_7 - 2x_8 \ge 0,$
$8x_1 - 2x_2 - 5x_9 + 2x_{10} + 12 \ge 0,\ -5x_1^2 - 8x_2 - (x_3 - 6)^2 + 2x_4 + 40 \ge 0,$
$-3(x_1 - 2)^2 - 4(x_2 - 3)^2 - 2x_3^2 + 7x_4 + 120 \ge 0,$
$-x_1^2 - 2(x_2 - 2)^2 + 2x_1x_2 - 14x_5 + 6x_6 \ge 0,$
$-0.5(x_1 - 8)^2 - 2(x_2 - 4)^2 - 3x_5^2 + x_6 + 30 \ge 0,$
$3x_1 - 6x_2 - 12(x_9 - 8)^2 + 7x_{10} \ge 0,\ -10 \le x_i \le 10,\ i = 1, \ldots, 10.$

each test function was performed in 30 independent runs, and data collection included (1) the best value, (2) the mean function value, and (3) the standard deviation of the function values.

Table 1 shows that the test functions involved 13, 8, 7, or 10 variables (factors), which required 13, 8, 7, or 10 columns, respectively, to allocate them in the Latin square used in the LSQEA approach. The Latin square $L_8(2^7)$ was used for 7 variables because it had 7 columns. The Latin square $L_{16}(2^{15})$ was used for 13, 8, or 10 variables because it had 15 columns. In this case, the first 13, 8, or 10 columns were used, whereas the remaining 2, 7, or 5 columns, respectively, were ignored. The computational procedures and evolutionary environments used to solve the test functions by QGA and RGA approaches were the same as those used in the LSQEA approach. However, Latin square was not used in QGA and RGA, and quantum-inspired computing was not used in the RGA.

In Table 2, the comparison of results obtained by LSQEA, QGA, and RGA approaches reveals the following.

(1) The proposed LSQEA finds optimal or near-optimal solutions.

(2) For all test functions, the LSQEA solutions are closer to optimal compared to the QGA and RGA solutions.

Table 2: Results of performance comparisons of LSQEA, QGA, and RGA.

Test function	Best value			Mean function value (standard deviation)			Globally minimal function value
	LSQEA	QGA	RGA	LSQEA	QGA	RGA	
$f_1(X)$	−15.000	−15.000	−15.000	−14.998 (0.005)	−14.997 (0.007)	−14.988 (0.055)	−15.000
$f_2(X)$	7117.961	7135.440	7178.168	7390.259 (197.317)	7614.255 (322.821)	7791.224 (480.058)	7049.331
$f_3(X)$	680.630	680.660	680.804	680.772 (0.104)	681.459 (0.836)	681.515 (0.891)	680.630
$f_4(X)$	24.306	24.462	24.694	24.981 (0.479)	25.980 (0.886)	26.051 (1.697)	24.306

(3) For all test functions, the deviations in function values are smaller in the proposed LSQEA than in the QGA and RGA. That is, the proposed LSQEA has a relatively more stable solution quality. Since the RGA is largely based on stochastic search techniques, the standard deviations in all evaluations of test functions are higher in the RGA than in the LSQEA and QGA.

Figure 1 shows convergence results on test functions f_1, f_2, f_3, and f_4 by using the LSQEA, QGA, and RGA. The LSQEA requires fewer function calls to reach the best value and has the sharper decline than the QGA and GA. That is, the LSQEA has faster convergence speed than the QGA and GA.

In the computational experiment by using the systematic reasoning ability of Latin squares, it was confirmed that a new individual generated by each matrix experiment is superior to its parents, two Q-bit individuals. That is, potential individuals in microspace can be exploited. In micro Q-bit space, the systematic reasoning mechanism of the Latin square with signal-to-noise ratio enhanced the performance of the LSQEA by accelerating convergence to the global solution. In macro Q-bit space, quantum-inspired computing with the GA enhanced the performance of the LSQEA. Table 2 shows that the QGA outperformed the RGA, which indicates that quantum-inspired computing with GA improves the performance of the QGA. Therefore, the LSQEA outperforms the QGA and RGA methods in both exploration and exploitation.

García et al. [54, 55] confirmed the use of the most powerful nonparametric statistical tests to carry out multiple comparisons. Therefore, this study used the nonparametric Wilcoxon matched-pairs signed-rank test [56] to tackle a multiple-problem analysis to compare two algorithms over a set of problems simultaneously. Let d_i be the difference between the performance scores of the two algorithms on ith out of n different runs. The differences are ranked according to their absolute values, and average ranks are assigned in case of ties. Let T^+ be the sum of ranks for the different runs on which the second algorithm outperformed the first, and let T^- be the sum of ranks for the opposite. Ranks of $d_i = 0$ are split evenly among the sums, and if there is an odd number of them, one is ignored

$$T^+ = \sum_{d_i>0} \operatorname{rank}(d_i) + \frac{1}{2}\sum_{d_i=0} \operatorname{rank}(d_i), \qquad T^- = \sum_{d_i<0} \operatorname{rank}(d_i) + \frac{1}{2}\sum_{d_i=0} \operatorname{rank}(d_i). \tag{4.1}$$

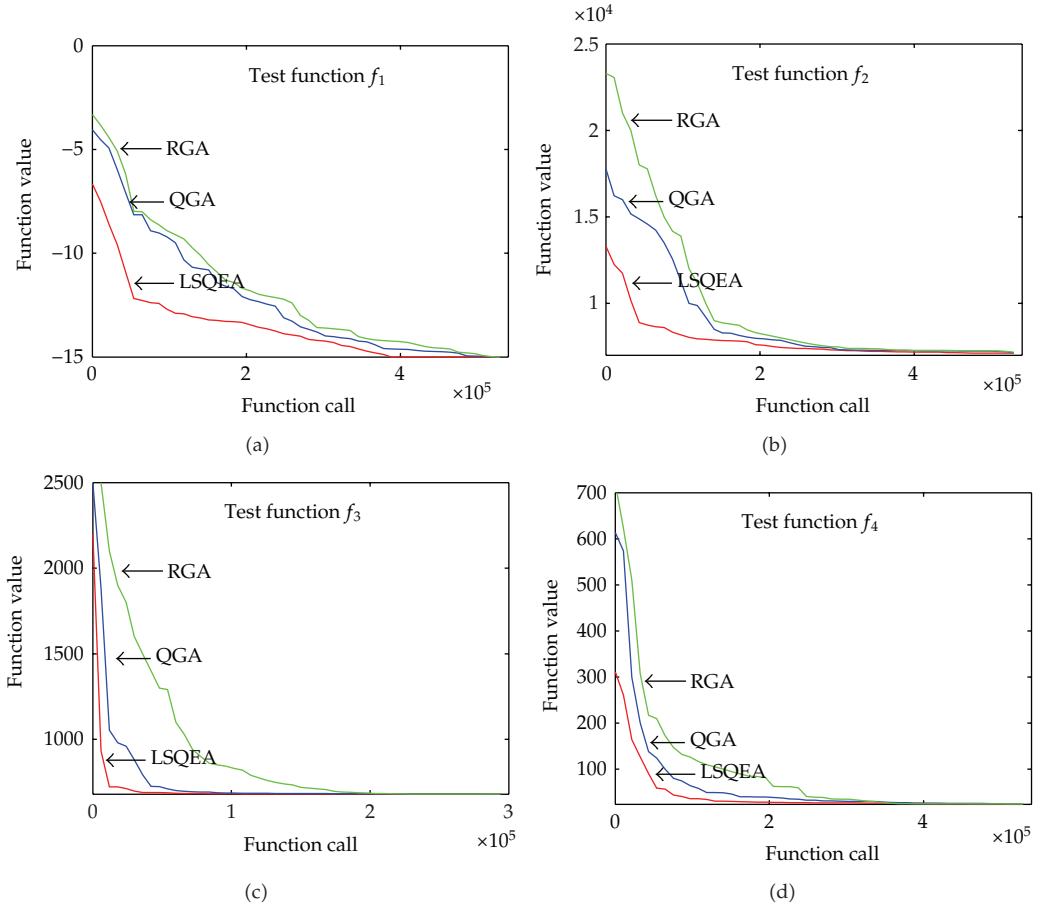

Figure 1: Convergence results on test functions f_1, f_2, f_3, and f_4 by using the LSQEA, QGA, and RGA.

Let T be the smaller of the two values, $T = \min(T^+, T^-)$. If T is less than or equal to the value of the distribution of Wilcoxon for n degrees of freedom, the null hypothesis of equality of means is rejected [54, 55]. Also, to calculate the significance of the test statistic (T), the mean ($\overline{T}$) and standard error (SE_T) were defined as follows [57]:

$$\overline{T} = \frac{n(n+1)}{4}, \qquad \mathrm{SE}_T = \sqrt{\frac{n(n+1)(2n+1)}{24}}, \tag{4.2}$$

where n is the sample size. Therefore, $Z = (T - \overline{T})/\mathrm{SE}_T$. If Z is bigger than 1.96 (ignoring the minus sign) then it is significant at $P < 0.05$.

The sample data obtained by Table 2 include the best values, mean function values, and standard deviation of mean function values. The Z values are all 2.934 ($P = 0.0033$) in LSQEA versus QGA, LSQEA versus RGA, and QGA versus RGA. So, the tests mean there is a significant difference between LSQEA and the two algorithms, QGA and RGA. That is, the performance of LSQEA really outperform those of QGA and RGA, since the Wilcoxon

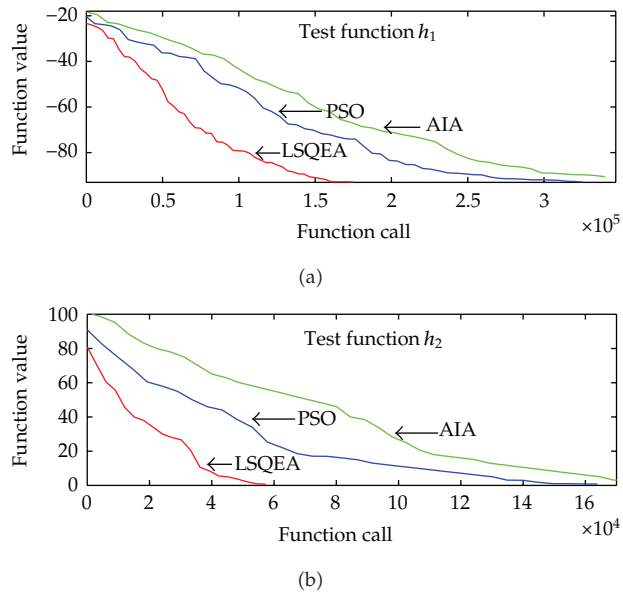

Figure 2: Convergence results on test functions h_1 and h_2 by using the LSQEA, PSO, and AIA.

test $Z > 1.96$ and $P < 0.05$. There is a significant difference between QGA and RGA. The performance of QGA is superior to that of RGA.

For evaluating the LSQEA in a problem which has a relatively larger dimensionality, two test examples which are 100 dimensions were used and minimized. They are $h_1 = -\sum_{i=1}^{n} \sin(x_i)\sin^{20}((i \times x_i^2)/\pi)$, $0 \le x_i \le \pi$, and $h_2 = \sum_{j=1}^{n-1}[100(x_j^2 - x_{j+1})^2 + (x_j - 1)^2]$, $-5 \le x_j \le 10$, where $n = 100$. To ensure a fair comparison of the performance of the LSQEA with that of recently proposed algorithms which are particle swarm optimization (PSO) [58] and artificial immune algorithm (AIA) [59], the study use the same population size that is 200 for 50 independent runs. The results of LSQEA on $h_1(x)$ in terms of mean function value (standard deviation) and mean function call (standard deviation) are −92.830 (0) and 178347 (14362), respectively, and on $h_2(x)$ are 0.7 (0) and 60377 (3368), respectively. The results of PSO on $h_1(x)$ in terms of mean function value (standard deviation) and mean function call (standard deviation) are −92.825 (0.03) and 330772 (29516), respectively, and on $h_2(x)$ are 0.752 (0.02) and 168736 (19325), respectively, while the results of AIA on $h_1(x)$ are −90.54 (0.93) and 346972 (29842), respectively, and on $h_2(x)$ are 2.95 (1.29) and 178048 (75619), respectively. Figure 2 shows convergence results on test functions h_1 and h_4 by using the LSQEA, PSO, and AIA. The LSQEA requires fewer function calls to reach the best value and has the sharper decline than the PSO and AIA. That is, the LSQEA has faster convergence speed than the PSO and AIA. In general, the performance of the LSQEA is superior to those of the PSO and AIA in the two examples. Additionally, the nonparametric Wilcoxon matched-pairs signed-rank test was used to evaluate the performance in two algorithms. The Wilcoxon test Z values are all 2.521 ($P = 0.0117$) in LSQEA versus PSO and LSQEA versus AIA. So, the tests mean there is a significant difference between LSQEA and the two algorithms, PSO and AIA. That is, the performance of LSQEA really outperforms those of PSO and AIA, since the Wilcoxon test $Z > 1.96$ and $P < 0.05$. There is also a significant difference between PSO and

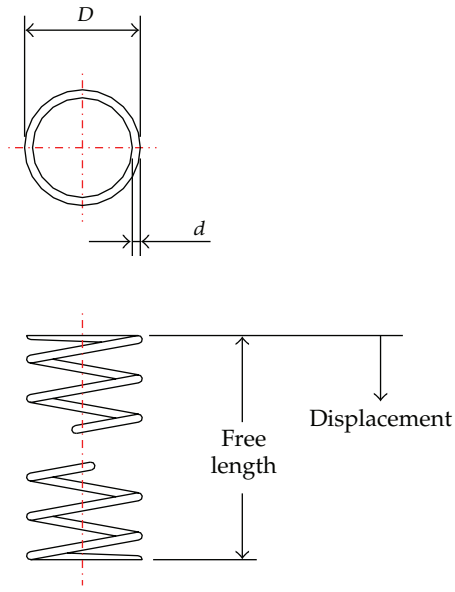

Figure 3: Compression coil spring.

AIA, since Z value is 2.521, and P value is 0.0117. The performance of PSO is superior to that of AIA in the two examples.

Hence, the performance improvement in the proposed LSQEA is achieved by using quantum-inspired computing and the systematic reasoning mechanism of Latin square with signal-to-noise ratio. Therefore, we conclude that the proposed LSQEA approach effectively solves the six nonlinear programming optimization problems with continuous variables. After confirming this capability of the LSQEA approach, the LSQEA approach was then evaluated for use in solving mixed discrete-continuous nonlinear design problems.

4.2. Solving Mixed Discrete-Continuous Nonlinear Design Problems

To evaluate the use of the LSQEA approach for solving mixed discrete-continuous nonlinear design problems in the engineering design field, this study applied the experimental design method reported in Chou et al. [53] for parameter adjustment in different evolutionary environments.

Example 4.1 (compression coil spring design). Figure 3 shows the first example, which is the design for a compression coil spring under a constant load for minimum volume of material. The relationships among the three design variables can be expressed as $X = [N, d, D]^T = [x_1, x_2, x_3]^T$, where N is an integer representing the number of coil springs; d is a discrete value representing the wire diameter according to ASTM code; D is a continuous variable representing winding diameter. As described by Sandgren [2], the objective and constraint equations can be mathematically derived as follows:

$$\text{minimize} \quad f(X) = \frac{\pi^2 x_3 x_2^2 (x_1 + 2)}{4}, \tag{4.3}$$

subject to

$$\begin{aligned}
g_1(X) &= S - \frac{8K_s P_{\max} x_3}{\pi x_2^3} \geq 0,\\
g_2(X) &= l_{\max} - (\delta + 1.05(x_1 + 2)x_2) \geq 0,\\
g_3(X) &= x_2 - d_{\min} \geq 0,\\
g_4(X) &= D_{\max} - x_3 \geq 0,\\
g_5(X) &= C - 3 \geq 0,\\
g_6(X) &= \delta_{p_m} - \delta \geq 0,\\
g_7(X) &= l_f - \delta - \frac{P_{\max} - P_{\text{load}}}{K} - 1.05(x_1 + 2)x_2 \geq 0,\\
g_8(X) &= \frac{P_{\max} - P_{\text{load}}}{K} - \delta_w \geq 0,
\end{aligned} \tag{4.4}$$

where

$$\begin{aligned}
C &= \frac{x_3}{x_2},\\
K_s &= (4C - 1)(4C - 4) + \frac{0.615}{C},\\
\delta &= \frac{8P_{\max} x_3^3 x_1}{G x_2^4},\\
K &= \frac{G x_2^4}{8 x_3^3 x_1},\\
5 \leq x_1 \leq 20, &\quad x_1 = 5 + k, \quad k = 0, 1, \ldots, 15,\\
x_2 \in \{&0.207, 0.225, 0.244, 0.263, 0.283, 0.307, 0.331, 0.362, 0.394, 0.4375, 0.500\},\\
1.0 \leq x_3 &\leq 3.0.
\end{aligned} \tag{4.5}$$

The assigned parameter values are $P_{\max} = 1000$, $S = 1.89 \times 10^5$, $G = 1.15 \times 10^7$, $l_{\max} = 14$, $d_{\min} = 0.2$, $D_{\max} = 3$, $\delta_{p_m} = 6$, $P_{\text{load}} = 300$, $l_f = 6.6$, and $\delta_w = 1.25$ [12].

The evolutionary environmental parameters applied in the computational experiments performed using the proposed LSQEA approach are p_s (population size) 100, p_c (crossover rate) 0.9, p_m (mutation rate) 0.3, and generation number 100. The design function is performed in 30 independent runs. Table 3 shows that the computational results obtained by the proposed LSQEA approach are superior to those obtained by the methods developed by Sandgren [2] and by Rao and Xiong [12]. Another observed advantage is that, unlike the approach presented in Sandgren [2], the LSQEA can use arbitrary starting points, which enhances its versatility and effectiveness. Table 4 further shows that the robustness analysis of

Table 3: Optimal design of compression coil spring.

Quantity	Best solutions			Remarks
	Branch and bound [2]	GA-based [12]	LSQEA	
x_1	10	9	9	Integer
x_2	0.283	0.283	0.283	Discrete
x_3	1.180701	1.22528	1.223042	Continuous
$f(X)$	2.7995	2.66342	2.658557	

Table 4: Robustness analysis results for compression coil spring design obtained by LSQEA.

Method	Min $f(X)$	Average $f(X)$	Standard deviation of $f(X)$
LSQEA	2.658557	2.672943	2.167×10^{-2}
GA based [12]	2.66342	NA	NA
Branch and bound [2]	2.7995	NA	NA

the LSQEA obtains a very small standard deviation in 30 independent runs, which confirms its robustness for designing a compression coil spring.

Example 4.2 (pressure vessel design). Figure 4 shows a compressed air storage tank consisting of a cylindrical pressure vessel capped by hemispherical heads at both ends [2]. The vessel design problem is formulated according to the ASME boiler and pressure vessel code. The relationships among the design variables can be expressed as $X = [T_s, T_h, R, L]^T = [x_1, x_2, x_3, x_4]^T$, where T_s is shell thickness, T_h is spherical head thickness, R is shell radius, and L is shell length. The purpose of the objective function is to minimize the total cost, including the cost of the material and the cost of forming and welding the pressure vessel. The problem can be modeled as

$$\text{minimize} \quad f(X) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.8621x_1^2x_3, \tag{4.6}$$

subject to

$$\begin{aligned} g_1(X) &= x_1 - 0.0193x_3 \geq 0, \\ g_2(X) &= x_2 - 0.00954x_3 \geq 0, \\ g_3(X) &= \pi x_3^2 x_4 + \frac{4}{3}\pi_3^3 - 750 \times 1728 \geq 0, \\ g_4(X) &= 240 - x_4 \geq 0, \\ g_5(X) &= x_1 - 1.1 \geq 0, \\ g_6(X) &= x_2 - 0.6 \geq 0, \end{aligned} \tag{4.7}$$

where T_s and T_h represent discrete values, integer multiples of 0.0625 inches, while R and L are continuous variables.

Table 5: Optimal design for pressure vessel.

Quantity	Best solutions					Remarks
	Sandgren [2]	Fu et al. [7]	Shih and Lai [60]	Rao and Xiong [12]	LSQEA	
x_1	1.125	1.125	1.125	1.1875	1.125	Discrete
x_2	0.625	0.625	0.625	0.625	0.625	Discrete
x_3	48.97	48.3807	47.448	61.4483	58.27	Continuous
x_4	106.72	111.7449	119.98	27.4037	43.9	Continuous
$f(X)$	7982.5	8048.619	8160.80	7284.02	7204.914	

Table 6: Robustness analysis of pressure vessel design obtained by LSQEA.

Method	Min $f(X)$	Average $f(X)$	Standard deviation of $f(X)$
LSQEA	7204.914	7277.262	43.543
Rao and Xiong [12]	7284.02	NA	NA
Shih and Lai [60]	8160.80	NA	NA
Fu et al. [7]	8048.619	NA	NA
Sandgren [2]	7982.5	NA	NA

In the computational experiments to evaluate the proposed LSQEA approach, population size p_s is 300, the crossover rate p_c is 0.9, the mutation rate p_m is 0.3, and the generation number is 200. The design function was performed in 30 independent runs. Table 5 compares the computational results obtained by the proposed LSQEA approach and by the methods introduced by Sandgren [2], Fu et al. [7], Rao and Xiong [12], and Shih and Lai [60]. The comparison shows that the proposed LSQEA approach outperforms the methods developed by Sandgren [2], Fu et al. [7], Rao and Xiong [12], and Shih and Lai [60]. Additionally, unlike the methods developed by Sandgren [2], Fu et al. [7], Rao and Xiong [12], and Shih and Lai [60], LSQEA approach can use arbitrary starting points, which enhances its versatility and effectiveness. Table 6 further shows the results of the robustness analysis of the LSQEA. The standard deviation obtained in 30 independent runs is small, and the average value is better than the best values obtained by the methods reported in Sandgren [2], Fu et al. [7], Rao and Xiong [12], and Shih and Lai [60].

Example 4.3 (welded beam design). The objective in this example, which was given in Ragsdell and Phillips [61], is to find the structural welded beam design with the lowest cost (Figure 5). The considered constraints are weld stress, buckling load, beam deflection, and beam bending stress. The relationships among the design variables can be expressed as $X = [t, b, h, l]^T = [x_1, x_2, x_3, x_4]^T$, where t is bar thickness, b is bar breadth, h is weld thickness, and l is weld length. The objective of the problem is to minimize design cost. The major cost components of such a welded beam include the setup labor cost, welding labor cost, and material cost. The objective function can be described as

$$f(X) = (1 + c_1)x_3^2 x_4 + c_2 x_1 x_2 (L + x_4). \tag{4.8}$$

The following behavior constraints are considered [12].

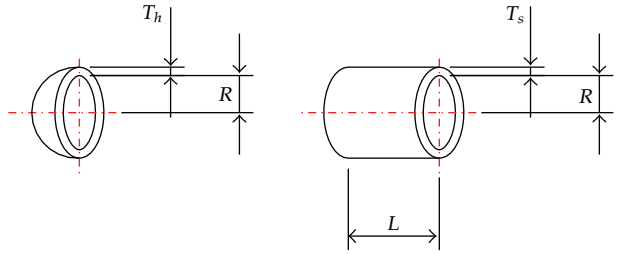

Figure 4: Pressure vessel.

(a) Upper bound of maximum shear stress $\tau(X)$ on the weld:

$$\tau(X) \leq \tau_d \tag{4.9}$$

with

$$\begin{aligned}
\tau(X) &= \left[(\tau')^2 + 2\tau'\tau'' \cos\theta + (\tau'')^2\right]^{1/2}, \\
\tau' &= \frac{F}{\sqrt{2}x_3x_4}, \\
\tau'' &= \frac{MR}{J}, \\
M &= \frac{F}{L + x_4/2}, \\
R &= \left\{\frac{x_4^2}{4} + \left[\frac{(x_1 + x_3)}{2}\right]^2\right\}^{1/2}, \\
J &= 2 \times 0.707x_3x_4\left\{\frac{x_4^2}{12} + \left[\frac{(x_1 + x_3)}{2}\right]^2\right\}, \\
\cos\theta &= \frac{x_4}{2R}.
\end{aligned} \tag{4.10}$$

(b) Upper bound of maximum normal stress $\sigma(X)$ on the beam:

$$\sigma(X) \leq \sigma_d \tag{4.11}$$

with

$$\sigma(X) = \frac{6FL}{x_2x_1^2}. \tag{4.12}$$

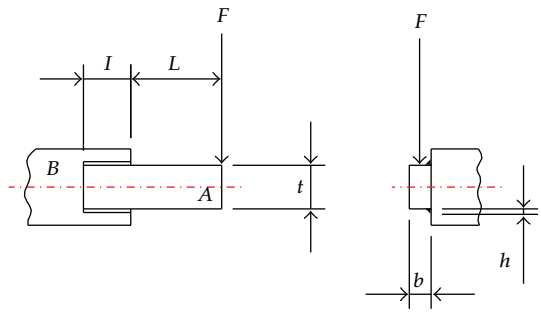

Figure 5: Welded beam.

(c) Lower bound of bulking load $P_c(x)$ on the beam:

$$P_c(X) \geq F \tag{4.13}$$

with

$$\begin{aligned} P_c(X) &= \frac{4.013\sqrt{EI\alpha}}{L^2}\left[1 - \frac{x_1}{2L}\sqrt{\frac{EI}{\alpha}}\right], \\ I &= \frac{x_1 x_2^3}{12}, \\ \alpha &= \frac{G x_1 x_2^3}{3}. \end{aligned} \tag{4.14}$$

(d) Upper bound of end deflection DEL(x) on the beam:

$$\mathrm{DEL}(X) \leq \delta_d \tag{4.15}$$

with

$$\mathrm{DEL}(X) \leq \frac{4FL^3}{Ex_1^3 x_2}. \tag{4.16}$$

Additionally, the side constraints, which are expressed as $x_3 \leq x_2$ and $x_3 \geq 0.125$, are considered along with the following numerical data [12]:

$$\begin{aligned} &c_1 = 0.37 \times 0.283, \quad c_2 = 0.17 \times 0.283, \quad L = 14, \quad F = 6000, \\ &\tau_d = 13600, \quad \sigma_d = 30000, \quad \delta_d = 0.25, \quad E = 3 \times 10^7, \quad G = 12 \times 10^6. \end{aligned} \tag{4.17}$$

The variables t and b are considered discrete variables (integer multiples of 0.5); h and l are considered integers.

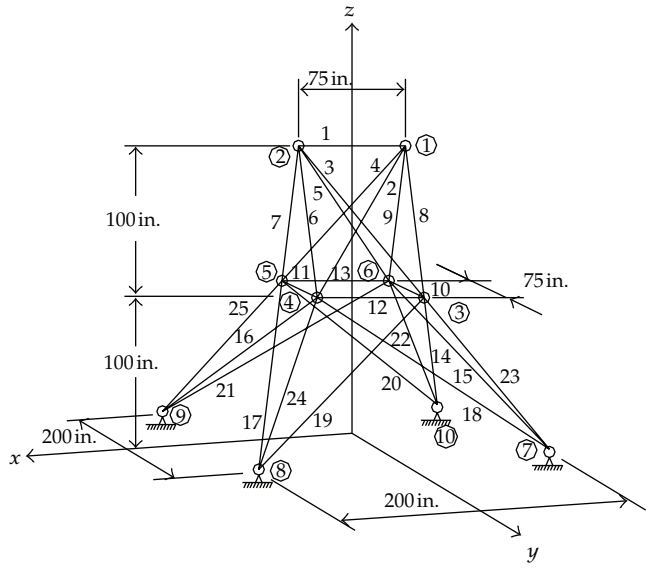

Figure 6: Twenty-five bar truss.

Table 7: Robustness analysis of welded beam design obtained by LSQEA.

Method	Min $f(X)$	Average $f(X)$	Standard deviation of $f(X)$
LSQEA	5.67334	5.67334	0
Rao and Xiong [12]	5.67334	NA	NA

The parameters used in the computational experiments performed to evaluate the proposed LSQEA approach are population size p_s of 10, crossover rate p_c of 0.9, mutation rate p_m of 0.3, and generation number of 20. The design function was performed in 30 independent runs. The computational results obtained by the proposed LSQEA approach are comparable to those obtained by Rao and Xiong [12]. The minimum cost is 5.67334, and $[t, b, h, l]^T = [4.5, 1.0, 1, 2]^T$. Table 7 also shows the results of robustness analysis of the LSQEA. The standard deviation of 0 obtained in 30 independent runs indicates that the LSQEA finds the optimal solution each run. That is, the LSQEA is a very robust and stable method for designing the welded beam. Additionally, the solution space in this welded beam design problem is small since only 4 design variables are used, and all are discrete variables (integer multiples and integers). Therefore, the LSQEA easily finds the optimal solution.

Example 4.4 (twenty-five bar truss design). Figure 6 shows Example 4.4, a twenty-five bar truss which is a classic test case often used to test optimization algorithms for both continuous and discrete structural optimization problems. Table 8 gives the two load conditions for this truss, which is designed under constraints on both member stress and Euler buckling stress. Since the truss is symmetrical, the member areas can be divided into eight groups: $A_1, A_2 = A_3 = A_4 = A_5$, $A_6 = A_7 = A_8 = A_9$, $A_{10} = A_{11}$, $A_{12} = A_{13}$, $A_{14} = A_{15} = A_{16} = A_{17}$, $A_{18} = A_{19} = A_{20} = A_{21}$, and $A_{22} = A_{23} = A_{24} = A_{25}$, thus eight independent areas are selected as continuous or discrete design variables. Three objective functions are considered in this

Table 8: Load conditions for twenty-five bar truss.

Joint	1	2	3	6
		Load condition 1 (pounds)		
F_x	0	0	0	0
F_y	20,000	−20,000	0	0
F_z	−5,000	−5,000	0	0
		Load condition 2 (pounds)		
F_x	1,000	0	500	500
F_y	10,000	10,000	0	0
F_z	−5,000	−5,000	0	0

example: (i) minimization of weight, (ii) minimization of deflection on nodes 1 and 2, and (iii) maximization of the fundamental natural frequency of vibration in the truss. The objective functions are as follows [12]:

$$\begin{aligned} \text{minimize} \quad & f_1(\mathrm{X}) = \sum_{i=1}^{25} \rho A_i l_i \,, \\ & f_2(\mathrm{X}) = \left(\delta_{1x}^2 + \delta_{1y}^2 + \delta_{1z}^2\right)^{1/2} + \left(\delta_{2x}^2 + \delta_{2y}^2 + \delta_{2z}^2\right)^{1/2}, \\ & f_3(\mathrm{X}) = \frac{1}{\omega_1}, \end{aligned} \tag{4.18}$$

where l_i denotes the length of the member i; $\rho = 0.1\,\mathrm{lb/in^3}$ is the weight density; δ_{ix}, δ_{iy}, and δ_{iz} are the x, y, and z components, respectively, of deflections in node $i (i = 1, 2)$; ω_1 is the fundamental natural frequency of vibration. The constraints can be stated as

$$\begin{gathered} \left|\sigma_{ij}(\mathrm{X})\right| \le S \quad (i = 1, 2, \ldots, 25,\ j = 1,\ \ 2), \\ \sigma_{ij}(\mathrm{X}) \ge B_i(\mathrm{X}) \quad (i = 1,\ 2, \ldots, 25,\ j = 1,\ \ 2), \\ x_i^l \le x_i \le x_i^u \quad (i = 1, 2, \ldots, 8), \end{gathered} \tag{4.19}$$

where σ_{ij} denotes the tension or compression stress in member i under load condition j; allowable stress S is set to 40000 psi; x_i^l and x_i^u are the lower and upper bounds of x_i, which are set to be $0.1\,\mathrm{in^2}$ and $5.0\,\mathrm{in^2}$, respectively; the Euler bulking stress $B_i(X)$ in member i is

$$B_i(\mathrm{X}) = \frac{-100.01\pi E A_i}{8 l_i^2} \quad (i = 1, 2, \ldots, 25), \tag{4.20}$$

in which E is the Young modulus of 1.0×10^7 psi.

If each x_i is considered a discrete variable, the following expression is used: $x_i = 0.1 + 0.1k$, where $k = 0, 1, 2, \ldots, 49$. The length of each bar is calculated according to the coordinates of the nodes of the truss. The deflections (δ_{ix}, δ_{iy}, and δ_{iz}) of node i, the fundamental natural vibration frequency ω_1, and the stress σ_{ij} in member i under the load condition j are obtained by finite-element analysis of the truss with ANSYS CAE Toolbox [62].

Table 9: Best results for individual objective functions obtained by LSQEA.

Quantity	Min. weight	Min. deflection	Max. frequency
Continuous results			
x_1	0.1	3.456	0.1000
x_2	0.8023	5.0	0.7880
x_3	0.7479	5.0	0.7538
x_4	0.1	3.3183	0.9000
x_5	0.1245	5.0	0.1001
x_6	0.5711	5.0	4.8713
x_7	0.9783	5.0	2.8019
x_8	0.8026	5.0	5.0000
ANSYS CAE Toolbox			
Weight (lb.)	233.0609	1616.798	911.9918
Deflection (in.)	1.9271	0.3085	1.2854
Frequency (Hz)	73.4297	70.7414	113.8128
Discrete results			
x_1	0.1	1.4	0.1
x_2	0.9	5.0	0.8
x_3	1.0	5.0	0.8
x_4	0.1	2.1	0.9
x_5	0.1	4.8	0.1
x_6	0.5	5.0	4.8
x_7	0.9	5.0	2.9
x_8	1.0	5.0	5.0
ANSYS CAE Toolbox			
Weight (lb.)	248.2764	1580.1035	916.5322
Deflection (in.)	1.6542	0.3086	1.2371
Frequency (Hz)	73.1060	73.1025	113.2998

Table 9 shows the best continuous and discrete results obtained by the proposed LSQEA approach with ANSYS CAE Toolbox. In contrast, Table 10 shows the best results obtained by the Rao and Xiong [12] approach using the ANSYS CAE Toolbox. The comparison of results in Tables 9 and 10 confirm that the proposed LSQEA approach provides better results in each main objective function compared to the method developed by Rao and Xiong [12]. Meanwhile, for each specified design objective function, the other two objectives are also better than those reported in Rao and Xiong [12]. An interesting question is why the discrete results given in Rao and Xiong [12] achieve a higher optimal natural frequency compared to the continuous results (Table 10). Intuitively, the answers found in a continuous space should be best. That is, in the case in which the natural frequency is maximized, the results presented by Rao and Xiong [12] are inapplicable. Another issue is the three individual objective functions, which are apparently dependent. Reducing the weight of the truss requires an increase in deflection and a decrease in the fundamental natural frequency, and vice versa. Although reaching the global optimum is possible in single-objective optimization problems, it is reached at the expense of performance in achieving the other two objectives. Therefore, the reasonable design specification for a practical

Table 10: Best results for individual objective functions [12].

Quantity	Min. weight	Min. deflection	Max. frequency
Continuous results			
x_1	0.1	3.7931	0.1
x_2	0.8023	5.0	0.7977
x_3	0.7479	5.0	0.7461
x_4	0.1	3.3183	0.7282
x_5	0.1245	5.0	0.8484
x_6	0.5712	5.0	1.9944
x_7	0.9785	5.0	1.9176
x_8	0.8025	5.0	4.1119
ANSYS CAE Toolbox			
Weight (lb.)	233.0727	1619.3258	600.8789
Deflection (in.)	1.9271	0.3085	1.3565
Frequency (Hz)	73.4255	70.3725	108.8761
Discrete results			
x_1	0.1	3.0	0.1
x_2	0.9	5.0	0.8
x_3	0.9	5.0	0.8
x_4	0.1	3.0	0.9
x_5	0.2	4.8	0.1
x_6	0.6	5.0	4.8
x_7	1.0	5.0	2.9
x_8	0.8	5.0	5.0
ANSYS CAE Toolbox			
Weight (lb.)	249.3187	1605.6035	916.5322
Deflection (in.)	1.7542	0.3086	1.2371
Frequency (Hz)	70.0243	71.2641	113.2998

engineering application is an essential consideration. If the defined objective is maximizing the fundamental natural frequency of vibration in the truss subject to the limited deflections in nodes 1 and 2, reasonable results are obtained. In this example, the results in the third case (maximizing the fundamental natural frequency of vibration in the truss) show the design characteristics.

In summary, the above results confirm that the LSQEA approach obtains robust and stable results. Therefore, we conclude that the LSQEA is highly feasible for solving the four examples that are mixed-discrete-continuous nonlinear design optimization problems.

5. Conclusions

The LSQEA approach proposed in this study solves the problems of mixed discrete-continuous nonlinear design optimization. The approach combines the merit of QGA, which is its powerful global exploration capability for exploring the optimal feasible region, with that of the Latin square, which is exploitation of the better offspring. In this study, the LSQEA approach efficiently solved global numerical optimization problems with continuous

variables. The computational experiments show that, compared to QGA and RGA, the proposed LSQEA is more efficient in finding optimal or near-optimal solutions for nonlinear programming optimization problems with continuous variables. Additional advantages of the LSQEA approach include its superior robustness compared to QGA and RGA and its global exploration capability. Finally, applications of the LSQEA to solve various mixed discrete-continuous nonlinear design optimization problems in computational experiments in this study confirm its superior solution quality and superior robustness compared to existing methods. Therefore, the proposed LSQEA approach can be used as a global optimization method for solving mixed discrete-continuous nonlinear design problems complicated by multiple constraints.

Acknowledgment

This work was partially supported by the National Science Council, Taiwan, under Grant no. NSC96-2221-E-153-002-MY2, NSC99-2221-E-151-071-MY3, NSC100-2221-E-153-001, NSC101-2221-E-153-003, and NSC 101-2320-B-037-022.

References

[1] P. Hajela and C. J. Shih, "Optimal design of laminated composites using a modified mixed integer and discrete programming algorithm," *Computers and Structures*, vol. 32, no. 1, pp. 213–221, 1989.
[2] E. Sandgren, "Nonlinear integer and discrete programming in mechnical design optimization," *ASME Journal of Mechanical Design*, vol. 112, no. 2, pp. 223–229, 1990.
[3] J. S. Arora, M. W. Huang, and C. C. Hsieh, "Methods for optimization of nonlinear problems with discrete variables: a review," *Structural Optimization*, vol. 8, no. 2-3, pp. 69–85, 1994.
[4] M. Bremicker, P. Y. Papalambros, and H. T. Loh, "Solution of mixed-discrete structural optimization problems with a new sequential linearization algorithm," *Computers and Structures*, vol. 37, no. 4, pp. 451–461, 1990.
[5] H. T. Loh and P. Y. Papalambros, "Sequential linearization approach for solving mixed-discrete nonlinear design optimization problems," *ASME Journal of Mechanical Design*, vol. 113, no. 3, pp. 325–334, 1991.
[6] D. K. Shin, Z. Gurdal, and O. H. Grin, "A penalty approach for nonlinear optimization with discrete design variables," *Engineering Optimization*, vol. 16, pp. 29–42, 1990.
[7] J. F. Fu, R. G. Fenton, and W. L. Cleghorn, "A mixed integer-discrete continuous programming method and its application to engineering design optimization," *Engineering Optimization*, vol. 17, pp. 263–280, 1991.
[8] J. Cai and G. Thieraut, "Discrete optimization of structures using an improved penalty function method," *Engineering Optimization*, vol. 17, pp. 293–306, 1993.
[9] O. Jonsson and T. Larsson, "Lagrangean relaxation and sub-gradient optimization applied to optimal design with discrete sizing," *Engineering Optimization*, vol. 16, pp. 221–233, 1990.
[10] S. S. Lin, C. Zhang, and H. P. Wang, "On mixed-discrete nonlinear optimization problems: a comparative study," *Engineering Optimization*, vol. 23, pp. 287–300, 1995.
[11] S. J. Wu and P. T. Chow, "Applications of genetic algorithms to discrete optimization problems," *Journal of the Chinese Society of Mechanical Engineers*, vol. 16, no. 6, pp. 587–598, 1995.
[12] S. S. Rao and Y. Xiong, "A hybrid genetic algorithm for mixed-discrete design optimization," *Journal of Mechanical Design*, vol. 127, no. 6, pp. 1100–1112, 2005.
[13] W. Tang and Q. Yuan, "Improved genetic algorithm for shape optimization of truss structures," *Chinese Journal of Theoretical and Applied Mechanics*, vol. 38, no. 6, pp. 843–849, 2006.
[14] R. L. Haupt, "Antenna design with a mixed integer genetic algorithm," *IEEE Transactions on Antennas and Propagation*, vol. 55, no. 3, pp. 577–582, 2007.
[15] K. Deep, K. P. Singh, M. L. Kansal, and C. Mohan, "A real coded genetic algorithm for solving integer and mixed integer optimization problems," *Applied Mathematics and Computation*, vol. 212, no. 2, pp. 505–518, 2009.

[16] K. M. Lee, J. T. Tsai, T. K. Liu, and J. H. Chou, "Improved genetic algorithm for mixed-discrete-continuous design optimization problems," *Engineering Optimization*, vol. 42, no. 10, pp. 927–941, 2010.

[17] W. H. Ho and C. S. Chang, "Genetic-algorithm-based artificial neural network modeling for platelet transfusion requirements on acute myeloblastic leukemia patients," *Expert Systems with Applications*, vol. 38, no. 5, pp. 6319–6323, 2011.

[18] W. H. Ho, J. X. Chen, I. N. Lee, and H. C. Su, "An ANFIS-based model for predicting adequacy of vancomycin regimen using improved genetic algorithm," *Expert Systems with Applications*, vol. 38, no. 10, pp. 13050–13056, 2011.

[19] C. Zhang and H. P. Wang, "Mixed-discrete nonlinear optimization with simulated annealing," *Engineering Optimization*, vol. 21, pp. 277–291, 1993.

[20] W.-H. Ho, J.-H. Chou, and C.-Y. Guo, "Parameter identification of chaotic systems using improved differential evolution algorithm," *Nonlinear Dynamics*, vol. 61, no. 1-2, pp. 29–41, 2010.

[21] W.-H. Ho and A. L.-F. Chan, "Hybrid Taguchi-differential evolution algorithm for parameter estimation of differential equation models with application to HIV dynamics," *Mathematical Problems in Engineering*, vol. 2011, Article ID 514756, 14 pages, 2011.

[22] Y. J. Cao, L. Jiang, and Q. H. Wu, "An evolutionary programming approach to mixed-variable optimization problems," *Applied Mathematical Modelling*, vol. 24, no. 12, pp. 931–942, 2000.

[23] P. W. Shor, "Algorithms for quantum computation: discrete logarithms and factoring," in *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, pp. 124–134, Santa Fe, NM, USA, 1994.

[24] L. K. Grover, "Fast quantum mechanical algorithm for database search," in *Proceedings of the 28th Annual ACM Symposium on the Theory of Computing*, pp. 212–219, New York, NY, USA, May 1996.

[25] L. K. Grover, "Quantum mechanics helps in searching for a needle in a haystack," *Physical Review Letters*, vol. 79, no. 2, pp. 325–328, 1997.

[26] K. H. Han, K. H. Park, C. H. Lee, and J. H. Kim, "Parallel quantum-inspired genetic algorithm for combinatorial optimization problem," in *Proceedings of IEEE Conference on Evolutionary Computation*, pp. 1422–1429, Seoul, Korea, May 2001.

[27] K. H. Han and J. H. Kim, "Genetic quantum algorithm and its application to combinatorial optimization problem," in *Proceedings of the Congress on Evolutionary Computation*, pp. 1354–1360, San Diego, Calif, USA, July 2000.

[28] K. H. Han and J. H. Kim, "Quantum-inspired evolutionary algorithm for a class of combinatorial optimization," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 6, pp. 580–593, 2002.

[29] K. H. Han and J. H. Kim, "Quantum-inspired evolutionary algorithms with a new termination criterion, Hε gate, and two-phase scheme," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 2, pp. 156–169, 2004.

[30] A. Malossini, E. Blanzieri, and T. Calarco, "Quantum genetic optimization," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 2, pp. 231–241, 2008.

[31] I. Grigorenko and M. E. Garcia, "Calculation of the partition function using quantum genetic algorithms," *Physica A*, vol. 291, pp. 463–470, 2001.

[32] J. A. Yang, B. Li, and Z. Zhuang, "Multi-universe parallel quantum genetic algorithm and its application to blind source separation," in *Proceedings of the International Conference on Neural Networks and Signal Processing (ICNNSP '03)*, pp. 393–398, Nanjing, China, December 2003.

[33] G. Zhang, W. Jin, and L. Hu, "A novel parallel quantum genetic algorithm," in *Proceedings of the 4th International Conference on Parallel and Distributed Computing, Applications and Technologies*, pp. 693–697, Chengdu, China, August 2003.

[34] C. Hui, Z. Jiashu, and Z. Chao, "Chaos updating rotated gates quantum-inspired genetic algorithm," in *Proceedings of the International Conference on Communications, Circuits and Systems*, pp. 1108–1112, Chengdu, China, June 2004.

[35] L. Wang, F. Tang, and H. Wu, "Hybrid genetic algorithm based on quantum computing for numerical optimization and parameter estimation," *Applied Mathematics and Computation*, vol. 171, no. 2, pp. 1141–1156, 2005.

[36] Q. Yang and S. Ding, "Methodology and case study of hybrid quantum-inspired evolutionary algorithm for numerical optimization," in *Proceedings of the 3rd International Conference on Natural Computation (ICNC '07)*, pp. 608–612, Haikou, China, August 2007.

[37] N. Li, P. Du, and H. Zhao, "Independent component analysis based on improved quantum genetic algorithm: application in hyperspectral images," in *Proceedings of IEEE International Geoscience and Remote Sensing Symposium (IGARSS '05)*, pp. 4323–4326, Seoul, Korea, July 2005.

[38] L. Abdesslem, M. Soham, and B. Mohamed, "Multiple sequence alignment by quantum genetic algorithm," in *Proceedings of the 20th International Parallel and Distributed Processing Symposium*, pp. 8–15, Rhodes Island, Greece, 2006.
[39] Z. Dong, Y. Huang, and P. Han, "Thermal process identification with radial basis function network based on quantum genetic algorithm," *Proceedings of the Chinese Society of Electrical Engineering*, vol. 28, no. 17, pp. 99–104, 2008.
[40] J. Gao and J. Wang, "A hybrid quantum-inspired immune algorithm for multiobjective optimization," *Applied Mathematics and Computation*, vol. 217, no. 9, pp. 4754–4770, 2011.
[41] M. S. Phadke, *Quality Engineering Using Robust Design*, Prentice-Hall, Upper Saddle River, NJ, USA, 1989.
[42] D. C. Montgomery, *Design and Analysis of Experiments*, John Wiley & Sons, New York, NY, USA, 1991.
[43] S. H. Park, *Robust Design and Analysis for Quality Engineering*, Chapman and Hall, London, UK, 1996.
[44] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, Boston, Mass, USA, 1989.
[45] M. Gen and R. Cheng, *Genetic Algorithms and Engineering Design*, John Wiley and Sons, New York, NY, USA, 1997.
[46] J. T. Tsai, T. K. Liu, and J. H. Chou, "Hybrid Taguchi-genetic algorithm for global numerical optimization," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 4, pp. 365–377, 2004.
[47] J. T. Tsai, J. H. Chou, and T. K. Liu, "Tuning the structure and parameters of a neural network by using hybrid Taguchi-genetic algorithm," *IEEE Transactions on Neural Networks*, vol. 17, no. 1, pp. 69–80, 2006.
[48] W. Hock and K. Schittkowski, *Test Examples for Nonlinear Programming Codes*, vol. 187 of *Lecture Notes in Economics and Mathematical Systems*, Springer, Berlin, Germany, 1981.
[49] C. A. Floudas and P. M. Pardalos, *Recent Advances in Global Optimization*, Princeton University Press, Princeton, NJ, USA, 1992.
[50] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, Springer, Berlin, Germany, 1994.
[51] J. J. Grefenstette, "Optimization of control parameters for genetic algorithms," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 16, no. 1, pp. 122–128, 1986.
[52] L. Davis, "Adapting operator probabilities in genetic algorithms," in *Proceedings of the International Conference on Genetic Algorithms (ICGA '89)*, pp. 61–69, San Mateo, Calif, USA, 1989.
[53] J. H. Chou, W. H. Liao, and J. J. Li, "Application of Taguchi-genetic method to design optimal grey-fuzzy controller of a constant turning force system," in *Proceedings of the 15th CSME Annual Conference*, pp. 31–38, Taiwan, 1998.
[54] S. García, A. Fernández, J. Luengo, and F. Herrera, "A study of statistical techniques and performance measures for genetics-based machine learning: accuracy and interpretability," *Soft Computing*, vol. 13, no. 10, pp. 959–977, 2009.
[55] S. García, D. Molina, M. Lozano, and F. Herrera, "A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC'2005 Special Session on Real Parameter Optimization," *Journal of Heuristics*, vol. 15, no. 6, pp. 617–644, 2009.
[56] F. Wilcoxon, "Individual comparisons by ranking method," *Biometrics*, vol. 1, pp. 80–83, 1945.
[57] A. Field, *Discovering Statistics Using SPSS*, SAGE Publications, London, UK, 2006.
[58] J. Kennedy, R. C. Eberhart, and Y. Shi, *Swarm Intelligence*, Morgan Kaufmann, San Francisco, Calif, USA, 2001.
[59] L. N. de Castro and J. Timmis, *Artificial Immune Systems: A New Computational Intelligence Approach*, Springer, London, UK, 2002.
[60] C. J. Shih and T. K. Lai, "Mixed-discrete fuzzy programming for nonlinear engineering optimization," *Engineering Optimization*, vol. 23, pp. 187–199, 1995.
[61] K. M. Ragsdell and D. T. Phillips, "Optimal design of a class of welded structure using geometric programming," *ASME Journal of Engineering for Industry-Transactions*, vol. 98, no. 3, pp. 1021–1025, 1976.
[62] ANSYS, *APDL Programmer's Guide: ANSYS Release 10.0*, ANSYS, Canonsburg, Pa, USA, 2005.