NOTICE: this is the author's version of a work that was accepted for publication in *Expert Systems with Applications*. Changes resulting from the publishing process, such as peer review, editing, corrections, structural formatting, and other quality control mechanisms may not be reflected in this document. Changes may have been made to this work since it was submitted for publication. A definitive version was subsequently published in Expert Systems with Applications, Vol. 37, no. 5 (2010) http://dx.doi.org/10.1016/j.eswa.2009.11.033

A new orthogonal array based crossover, with analysis of gene interactions, for evolutionary algorithms and its application to car door design

K.Y. Chan[1,2,*], C.K. Kwong[1], Y.C. Tsim[1], M.E. Aydin[2] and T.C. Fogarty[4]

[1]Department of Industrial and Systems Engineering,

The Hong Kong Polytechnic University,

Hung Hom, Kowloon,

Hong Kong, PRC

[2] Digital Ecosystems and Business Intelligence Institue, Curtin University of Technology, Perth, Australia

[3] Department of Computing and Information Systems,

University of Bedfordshire, Luton,

United Kingdom

[4] Faculty of Business, Computing and Information Management,

London South Bank University,

103 Borough Road, London,

United Kingdom

Abstract. Recent research shows that orthogonal array based crossovers outperform standard and existing crossovers in evolutionary algorithms in solving parametrical problems with high dimensions and multi-optima. However those crossovers employed so far, ignore the consideration of interactions between genes. In this paper, we propose a method to improve the existing orthogonal array based crossovers by integrating

information of interactions between genes. It is empirically shown that the proposed orthogonal array based crossover outperforms significantly both the existing orthogonal array based crossovers and standard crossovers on solving parametrical benchmark functions that interactions exist between variables. To further compare the proposed orthogonal array based crossover with the existing crossovers in evolutionary algorithms, a validation test based on car door design is used in which the effectiveness of the proposed orthogonal array based crossover is studied.

Keywords: crossover, evolutionary algorithms, interactions between genes, orthogonal array, car door design

1 Introduction

The approach of evolutionary algorithms is a type of stochastic searching method which is increasingly being used in a wide range of practical applications especially where the problem involves a non-differentiable cost function or where the cost function is hard to quantify mathematically [2, 6, 15, 34]. However, one of the main drawbacks of evolutionary algorithms is their tendency to converge before reaching an acceptable solution on challenging problems where the dimensions are high and there are numerous local optima [26, 44]. To overcome this problem, recent research [18, 19, 32, 45] has shown that optimization with evolutionary algorithms for solving parametrical problems with high dimensions and multi-optima can be enhanced by embedding the approach of orthogonal design in the crossover. Recent publications indicate that the orthogonal array based crossovers outperform the existing crossovers

* K.Y. Chan is the corresponding author of the paper. His email address is kit.chan@curtin.edu.au

in solving travelling salesman problems [18], polygonal approximation [24], solving multimedia multicast routing problems [51], searching Pareto-optimal solutions [31], development of fuzzy classifiers [20], structure-specified mixed H_2/H_∞ controller design [22], solving multiobjective combinatorial optimization problems [21], solving mesh optimization problems for surface approximation [25], process design of fluid dispensing [27], and PID controller design [9] in missile systems [16].

Orthogonal array based crossovers, where the salient feature is the incorporation of orthogonal design into the crossover, can be classified into two versions: The first version [31, 32, 51] is called the orthogonal crossover (OC), where chromosomes are produced by exploring alleles in parents based on combinations of an orthogonal array. The two top chromosomes with best fitness among all the chromosomes produced are selected as the two children as the outcome of the OC. However this approach only considers a limited number of combinations in the orthogonal array rather than taking the all the combinations as in a full factorial design. This may not be applicable for parametric problems, since the optimal combination may not be included in the combinations of the orthogonal array. The second version of orthogonal array based crossovers [9, 20-25, 27, 45], is called the main effect crossover (MC), because the combinations excluded in the orthogonal array are considered by analysing the main effects of genes in parents. Children are formed from the best combinations of genes with the best main effect in parents. Thus all combinations, as in full factorial design, are considered. This is more promising for parametric problems than OC on its own. It has been shown empirically that in general MC is better than OC in solving a set of parametrical benchmark problems with high dimensions [7, 9, 19].

MC allows us to approximate the main effect on each gene, but it ignores linkages in the form of interaction between genes. If strong interaction exists in localized features of the search space, misleading results may be obtained [14, 39-42]. It has been shown empirically that MC cannot achieve better results than OC on parametrical problems in which interactions exist between variables [7, 10]. In this paper, a new crossover operator called an interaction crossover (IC), that considers interactions between genes, is proposed. It employs the approach of the interaction plot [38], that has been commonly used to analysis interactions between parameters in industrial systems [29, 33, 35, 46, 50], to analyze the interaction between genes. From the interaction plot, a clear picture of the interaction effects between genes can be obtained. In the crossover operator the children can be produced by considering both the main effects in genes and interaction between genes. By solving a set of hard parametrical benchmark problems in which interactions exist between variables [48, 49], it has been shown empirically that significantly better results can be found based on IC compared with those based on OC, MC and standard crossovers.

To further evaluate the effectiveness of IC, a car door design based on Kim's work [Kim et al 2000], is optimized using the evolutionary algorithm with IC. The result is then compared with the results based on OC, MC and standard crossover and our previously developed computational method [3].

2 A Review of Orthogonal Array Based Crossovers

The following subsection 2.1 and 2.2 discuss the operations and the limitations of the two versions of orthogonal array based crossovers, orthogonal crossover OC [31, 32, 51] and main effect crossover MC [9, 20-25, 27, 45], respectively.

2.1 Orthogonal Crossover (OC)

In OC, an orthogonal array is integrated into the classical crossover operator so that two parents can be used to generate a small but representative set of sampling points, to be children based on the orthogonal array.

The chromosomes used in OC are in a real-coded representation, where the alleles in genes are real numbers. In a way similar to the classical crossover, two parents $P_1 = (p_{1,1}, p_{1,2},..., p_{1,l})$ and $P_2 = (p_{2,1}, p_{2,2},..., p_{2,l})$ are selected randomly from the population, where l is the number of variables in the chromosome.

Q is the number of levels, M is the number of rows and N is the number of columns of the orthogonal array $L_M(Q^N)$ respectively. Then the genes in P_1 and P_2 are quantified into Q levels such that the difference between any two successive levels is the same. The i^{th} level is denoted to be $Level(i) = (\beta_{i,1}, \beta_{i,2},...\beta_{i,l})$, where i=1,2,…,Q, and $\beta_{i,j}$ is defined as:

$$\beta_{i,j} = \begin{cases} \min(p_{1,j}, p_{2,j}), & \text{for } i = 1 \text{ and } 1 \le j \le l. \\ \min(p_{1,j}, p_{2,j}) + (i-1) \cdot \left(\dfrac{|p_{1,j} - p_{2,j}|}{Q-1} \right), & \text{for } 2 \le i \le Q-1 \text{ and } 1 \le j \le l \\ \max(p_{1,j}, p_{2,j}), & \text{for } i = Q \text{ and } 1 \le j \le l. \end{cases} \quad (1)$$

After quantifying P_1 and P_2, the Q levels are sampled as M potential offspring based on the combinations of the M rows of parameter levels in the orthogonal array $L_M(Q^N)$. Specifically, N-1 integers $k_1, k_2, \ldots, k_{N-1}$ are generated randomly such that $1 < k_1 < k_2 < \ldots < k_{N-1} < l$. Then N vectors are created such that:

$$\begin{cases} f^1 = \left(f_1^1, f_2^1, ..., f_{k_1}^1\right) = (1,2,...,k_1), \\ \quad \text{where the number of elements inside } f_1 \text{ is } k_1. \\ f^2 = \left(f_1^2, f_2^2, ..., f_{k_2-k_1+1}^2\right) = (k_1+1, k_1+2, ..., k_2), \\ \quad \text{where the number of elements inside } f_2 \text{ is } k_2 - k_1 + 1. \\ \vdots \qquad\qquad \vdots \qquad\qquad \vdots \\ \vdots \qquad\qquad \vdots \qquad\qquad \vdots \\ f^N = \left(f_1^N, f_2^N, ..., f_{l-k_{N-1}+1}^N\right) = (k_{N-1}+1, k_{N-1}+2, ..., l), \\ \quad \text{where the number of elements inside } f_N \text{ is } l - k_{N-1} + 1. \end{cases} \tag{2}$$

For i = 1,2,…M, the i^{th} offspring o_i is produced as:

$$o_i = (\beta_{a_1(i), f_1^1}, \beta_{a_1(i), f_2^1}, ..., \beta_{a_1(i), f_{k_1}^1}, \beta_{a_2(i), f_1^2}, \beta_{a_2(i), f_2^2}, ...$$

$$..., \beta_{a_2(i), f_{k_2-k_1+1}^2}, ..., \beta_{a_N(i), f_1^N}, \beta_{a_N(i), f_2^N}, ..., \beta_{a_N(i), f_{l-k_{N-1}+1}^N}) \tag{3}$$

where the combination of the i^{th} row of the orthogonal array $L_M\left(Q^N\right)$ is denoted as $a(i) = \left[a_1(i), a_2(i), ..., a_N(i)\right]$. The fitness c_i of each offspring $\left(\text{i.e.: } o_i \text{ for } i = 1,2,...,M\right)$ is evaluated according to the fitness function; i.e. $c_i = fun(o_i)$, where $fun(\ )$ denotes as the fitness function. Then the two offspring with the best fitness among M offspring are selected to be the two children of the OC.

Detailed steps of the OC are as follows:

Algorithm 1: Orthogonal crossover (OC)

Step 1: Select two parents P_1 and P_2 from the population randomly.

Step 2: Quantize P_1 and P_2 based on (1), and produce $Level(i)=(\beta_{i,1},\beta_{i,2},...\beta_{i,l})$, where i=1,2,…,Q.

Step 3: Randomly generate N-1 integers k_1 , k_2 ,…, k_{N-1} , such that $1< k_1 < k_2 <\ldots k_{N-1} <l$ in which l is the number of variables in the chromosomes. Then create N vectors f^i based on (2), where i=1,2,…,N.

Step 4: Apply $L_M(Q^N)$ to produce the M potential offspring o_i based on (3), where i=1,2,…,M.

Step 5: Evaluate the fitness of the M potential offspring based on the fitness function $fun(\ )$.

Step 6: Select the two offspring with the best fitness among M potential ones to be the two children of the OC.

The limitation of OC are described as follows: For i = 1,2,…,M, the i^{th} offspring o_i is produced based on the i^{th} combination of the orthogonal array $L_M(Q^N)$. Therefore M offspring are produced, meaning that M combinations are explored by the orthogonal array $L_M(Q^N)$. However, the total number of combinations of N genes with Q levels are N^Q. In OC, only M combinations are considered, thus $N^Q - M$ combinations are not explored. This may not be applicable

to some parametrical problems as the best combination may not be included in the orthogonal array. This is the potential limitation of this operator.

An example as shown in Figure 3 is used to explain the limitation in which the orthogonal array $L_4(2^3)$ [43] (which is detailed in Figure A1 in the appendix) is used to sample the genes from the two parents, P and P'. Each of the two parents are divided into three genes, where (p_1, p_2, p_3) and (p_1', p_2', p_3') are defined as the three genes of P and P' respectively. These three genes from the parents are sampled based on the four combinations of parameter levels in $L_4(2^3)$. Four potential offspring, O_1, O_2, O_3, and O_4 are produced as shown in Figure 3. The best two offspring among the four are selected to be the children of OC.

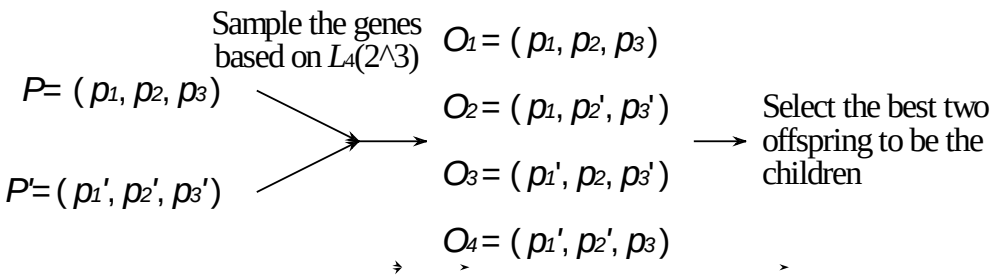

Figure 3 The orthogonal array $L_4(2^3)$ is used to sample the genes from P and P' for OC

Figure 4 illustrates the combinations in a full factorial design of 3 parameters with 2 levels. It can be seen from Figure 4 that the total number of combinations of 3 genes with 2 parameter levels is 8 (i.e. 2^3). However, only 4 combinations (the black points) are considered by the orthogonal array $L_4(2^3)$, and the other 4 combinations (the grey points) are not explored by $L_4(2^3)$. Therefore this approach may not be applicable to parametrical problems since the optimal combination may be one of the combinations of the grey points.

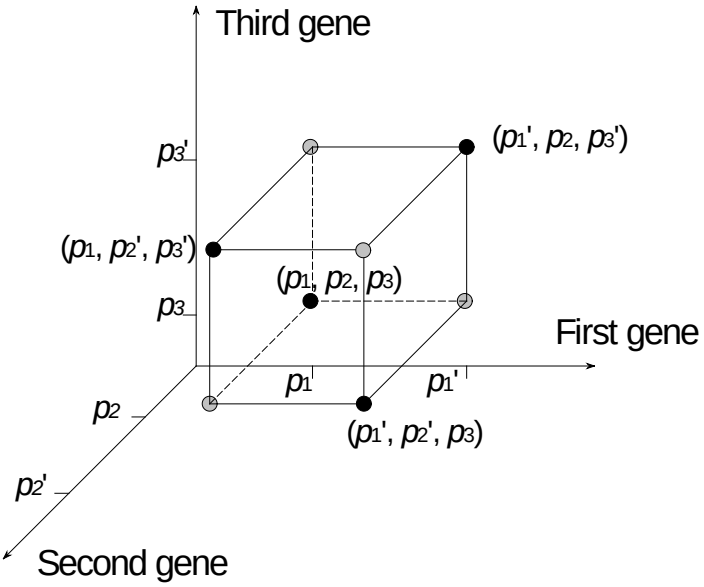

Figure 4 Combinations of the orthogonal array $L_4(2^3)$

2.2 Main Effect Crossover (MC)

The major steps of the main effect crossover (MC) [18] are similar to those of the OC. In the OC, after evaluating the fitness of the *M* offspring $\left(o_i \text{ for } i = 1,2,...M\right)$ the two top offspring among the offspring from the orthogonal array are selected to be the two children, after evaluating the fitness of the *M* offspring $\left(o_i \text{ for } i = 1,2,...M\right)$. In MC, the fitness values of the *M* offspring are analyzed further by considering the main effects in genes, and the children are produced by taking the genes with the best levels.

The main effect of the j^{th} gene with level *k* is defined as:

$$M_{jk} = \sum_{i=1}^{N} c_i \cdot \mathrm{H}_{ij} \tag{4}$$

where

$$\mathrm{H}_{ij} = \begin{cases} 1, \text{if } a_j(i) = k, \\ 0 \text{ otherwise}, \end{cases}$$

c_i is the fitness value of the i^{th} offspring o_i with i=1,2,…,M, i.e. $c_i = fun(o_i)$, and $a_j(i)$ is the element of the j^{th} column and the i^{th} row of the orthogonal array $L^M(Q^N)$. In other word, c_i represents the fitness value of the genes formed by the i^{th} combination of the orthogonal array $L^M(Q^N)$.

The first child is formed from the best combinations with the best level on each gene. For minimization problems, if $M_{j1} > M_{j2}$, the level 2 of the j^{th} gene is better than the level 1. The best level *Best(j)* of the j^{th} gene is denoted as:

$$Best(j) = \arg\left(\min_{k=1,2,...,Q} \left(M_{jk} \right) \right), \text{ where } j = 1,2,..., N \tag{5}$$

where 'arg(min(..))' is a function that returns the indices of the minimum value of the matrix. For maximization problem, if $M_{j1} > M_{j2}$, the level 1 of the j^{th} gene is better than the level 2. The best level *Best(j)* of the j^{th} gene is denoted as:

$$Best(j) = \arg\left(\max_{k=1,2,...,Q} \left(M_{jk} \right) \right), \text{ where } j = 1,2,..., N \tag{6}$$

where ' arg(max(..))' is a function that returns the indices of the maximum value of the matrix. The detailed description of the function 'arg' is referred to Example A1 in the appendix.

The second child is identical to the first child except that the gene with the lowest main effect difference at the other level is chosen, where the main effect difference $\left(MED_j \right)$ on the j^{th} gene is denoted as:

$$MED_j = \left| \max_{k=1,2,...,Q} \left(M_{jk} \right) - \min_{k=1,2,...,Q} \left(M_{jk} \right) \right|, \text{ where } j = 1,2,..., N. \tag{7}$$

Note that the main effect reveals the individual effect of a gene, thus the most effective gene has the largest main effect difference.

The detailed steps of the main effect crossover (MC) are as follows:

Algorithm 2: Main effect crossover (MC)

Step 1-Step 5: Step 1 to Step 5 are identical to Step 1 to Step 5 of Algorithm 1.

Step 6: Based on (4), evaluate the main effect M_{jk} of the j^{th} gene with level k, where j=1,2,…,N and k=1,2,…,Q.

Step 7: Determine the best level *Best(j)* of the j^{th} gene based on (5) for minimization problems or based on (6) for maximization problems, where j=1,2,…,N.

Step 8: The first child is formed from the best level of each gene.

Step 9: Determine the main effect difference $\left(MED_j\right)$ on the j^{th} gene based on (7), where j=1,2,…,N.

Step 10: The second child is identical to the first child except the gene with the lowest main effect difference adopts the other level.

The limitation of MC are described follows: In experimental design [1, 38], it should be emphasized that the analysis of the main effect is the simplest approach to data analysis. However, it is common for two of the genes to interact and yield a result that is more dependent upon the interaction between the two genes than on the main effects of either individual gene [14]. Further analysis, which gives insights into interactions and main effects inside the chromosomes in GAs, has been done [9, 39-42]. Their central idea is to perform an 'analysis of variance (ANOVA)', whereby the variability of the fitness values of the chromosomes (measured by sums of squared deviations from mean fitness, and denoted by *SS*) is partitioned into main effects and interactions; i.e.

Total *SS* = *SS* of main effects + *SS* of interactions

Therefore the lack of provision for adequately dealing with the potential interactions between genes is a major weakness of MC. If a chromosome exhibits very low interaction between the genes, it could probably be processed efficiently by MC. Otherwise the predicted optimal combination may not be reproducible if strong interaction exists between the genes.

Furthermore, the empirical results [7, 10] show that MC outperforms OC on the parametrical problems where all variables are linearly independent to each other. However, no significant improvement can be found on MC over OC on the parametrical problems where the variables interact with each other. Therefore, it seems that MC can not work well on parametrical problems in which variables interact with each other. In the following section, the improved version of MC, by integrating the information of interactions between genes, is proposed.

3 Interaction Crossover (IC)

The steps of the proposed new orthogonal array based crossover, namely IC, are similar to the ones in MC. In MC, the children are produced by considering only the best main effects in genes. In IC, the children are produced by considering both main effects in genes and interactions between genes. The approach of the interaction plot [38], which is commonly used to analyze the magnitudes of interaction between parameters in industrial systems [33, 35, 39, 46], is applied to IC. From the interaction plot, a clear picture of the magnitudes of interactions between genes can be indicated.

In IC, an interaction matrix MI_{ij} is prepared to estimate the magnitudes of interaction between gene i and j, where $1 \le i, j \le N$. It can be expressed as:

$$MI_{ij} = \left(I_{ij}(m,n); \text{for } 1 \le \text{m,n} \le \text{Q}\right)_{Q \times Q} \tag{8}$$

where Q is the number of rows and columns of the interaction matrix MI_{ij}. The elements in MI_{ij}, $I_{ij}(m,n)$, which represents the average fitness of the i^{th} gene with level m and j^{th} gene with level n, is defined as:

$$I_{ij}(m,n) = \frac{\sum_{p=1}^{N} f_p \cdot \left[\begin{array}{l} \text{the level of the } p^{th} \text{ offspring of the } i^{th} \text{ gene is } m \\ \text{and the } j^{th} \text{ gene is } n \end{array} \right]}{\sum_{p=1}^{N} \left[\begin{array}{l} \text{the level of the } p^{th} \text{ offspring of the } i^{th} \text{ gene is } m \\ \text{and the } j^{th} \text{ gene is } n \end{array} \right]} \qquad (9)$$

where $1 \le m, n \le Q$ and $\left[\text{condition}\right] = \begin{cases} 1 \text{ if the statement inside the bracket is true.} \\ 0 \text{ otherwise.} \end{cases}$

Then the approach of interaction plot [38] is used to indicate the magnitude of interaction between gene i and j. The r^{th} line of the interaction plot is defined as:

$$Line_{ij}(r) = \left(I_{ij}(1,r), I_{ij}(2,r), ..., I_{ij}(Q,r)\right), \text{ for } 1 \le r \le Q. \qquad (10)$$

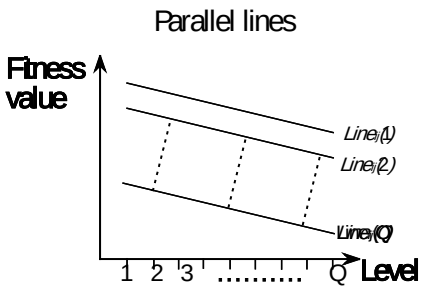

Figure 5(a) No interaction exists between gene i and j

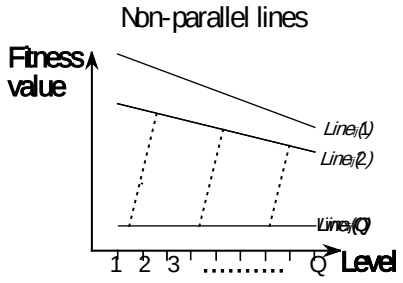

Figure 5(b) Interaction exists between gene i and j

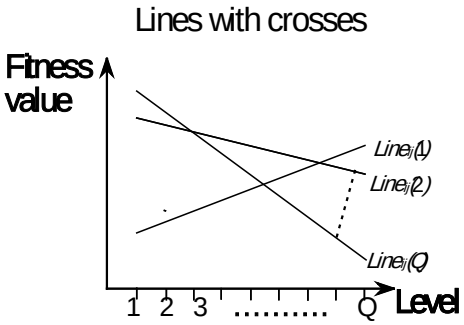

Figure 5(c) Strong interaction exists between gene *i* and *j*

The magnitude of interaction can be determined by the interaction plot. If the lines on the interaction plot (as shown in Figure 5(a)) are parallel, no interaction exists between gene *i* and *j*. If the lines on the interaction plots are nonparallel (as shown in Figure 5(b)), interaction occurs. If the lines cross (as shown in Figure 5(c)), strong interaction occurs. The actual amount of interaction between gene *i* and *j* can be determined by the number of intersections on the interaction plot.

If strong interaction does not exist in any of the gene pairs, then the main effects on genes can be separated out. The first child is formed by the combination of the genes with the best main effects based on (5) for minimization problems or on (6) for maximization problems. However, if strong interaction does exist in any one of the gene pairs, the first child is formed in two parts: The first part is the genes which do not carry any strong interaction between each other and the second part that in which the genes carry strong interaction between each other. In the first part, the level combination is formed by the genes with the best main effects based on (5) for minimization problems or on (6) for maximization problems. For the second part, the level combination of the genes, which gives the best fitness value, is chosen. Assume

that strong interaction exists between gene *i* and *j*. Then for minimization problems, the best level combination of gene *i* and *j* is given by:

$$[Best(i), Best(j)] = \arg\left(\min_{m,n=1,2,\ldots Q} \left(I_{ij}(m,n) \right) \right) \quad (11)$$

where $i, j = 1,2,\ldots N$ but $i \neq j$. 'arg(min(…))' is a function that returns the indices of the minimum value of the matrix. For maximization problems, the best level combination of gene *i* and *j* is given by:

$$[Best(i), Best(j)] = \arg\left(\max_{m,n=1,2,\ldots Q} \left(I_{ij}(m,n) \right) \right) \quad (12)$$

where $i, j = 1,2,\ldots N$ but $i \neq j$. 'arg(max(…))' is a function that returns the indices of the maximum value of the matrix. More detailed description of the function can be referred to Example A1 in the appendix.

If strong interaction exists both between gene *i* and *j* and between gene *j* and *k*, and the estimated interaction between *i* and *j* is larger than the one between *j* and *k*, then the gene pair of *i* and *j* will be selected, and the best level combination of gene *i* and *j* are given by equation (11) for minimization problems and (12) for maximization problems. Otherwise, the gene pair of *j* and *k* are selected, and the best level combination of gene *j* and *k* are given by equation (11) for minimization problems and (12) for maximization problems.

The second child is identical to the first child except that the gene with the lowest main effect difference in the other level is chosen. The main effect difference of the genes can be found by equation (7).

Detailed steps of IC are as follows:

Algorithm 3: Interaction crossover (IC)

Step 1- Step 6: Step 1 to Step 6 are identical to Step 1 to Step 6 of Algorithm 2.

Step 7: Construct the interaction matrix MI_{ij} by (8), where $i,j=1,2,\ldots N$ with $i \neq j$.

Step 8: Construct the interaction plot for MI_{ij} by using the lines given by (10), where $i, j=1,2,\ldots N$ with $i \neq j$.

Step 9: Identify whether the gene i and j holding strong interaction or not by checking whether any intersection exists on the interaction plot, where $i, j=1,2,\ldots N$ with $i \neq j$.

Step 10: The first child is formed by two parts. The first part is formed by the genes without carrying any strong interaction based on (5) for minimization problems and on (6) for maximization problems. The second part is formed by the genes that carry strong interaction based on (11) for minimization problems and on (12) for maximization problems.

Step 11: The second child is formed by performing Step 9 and Step 10 in Algorithm 2.

4 Numerical Result of Non-separable Benchmark Functions

We executed the evolutionary algorithms embedded with different crossovers to solve the benchmark functions ($f_1 - f_9$) shown in Table 1, which are all non-separable functions and the interactions existing between variables. $f_1 - f_6$ were collected from

[49] and $f_7 - f_9$ were collected from [48]. They are unlike separable functions in that each sub-function can be completely enumerated, thereby avoiding local optima that allow stochastic search methods to move the search into the basin of attraction of the global optimum of that sub-function. Also they cannot be decomposed into linear combinations of independent sub-functions since variables interact with each other and cannot be enumerated completely. They could be classified as good test suites for evolutionary algorithms since they are non-separable and each sub-function contains at least two variables [48].

Table 1 Non-separable benchmark functions

Test functions	Domain range (x_i)	Minimum
$f_1 = \min \sum_{i=1}^{N-1} \left[100\left(x_{i+1} - x_i^{\,2}\right)^2 + \left(x_i - 1\right)^2 \right]$	$[-5.12,5.12]^N$	0
$f_2 = \min\left(\sum_{i=1}^{N} \lvert x_i \rvert + \prod_{i=1}^{N} \lvert x_i \rvert \right)$	$[-10,10]^N$	0
$f_3 = \min \sum_{i=1}^{N} \left(\sum_{j=1}^{i} x_j \right)^2$	$[-100,100]^N$	0
$f_4 = \min\left(\frac{1}{4000} \sum_{i=1}^{N} x_i^{\,2} - \prod_{i=1}^{N} \cos\left(\frac{x_i}{\sqrt{i}} \right) + 1 \right)$	$[-600,600]^N$	0
$f_5 = \min\left\{ \sum_{i=1}^{N} 2N + \sum_{i=1}^{N-1} \left[\sin\left(x_i + x_{i+1}\right) + \sin\left(\frac{2x_i x_{i+1}}{3} \right) \right] \right\}$	$[3,13]^N$	0
$f_6 = \min\left(20 + e - 20e^{-0.2\sqrt{\frac{\sum_{i=1}^{N} x_i^{\,2}}{N}}} - e^{\sum_{i=1}^{N} \frac{\cos(20\pi x_i)}{N}} \right)$	$[-30,30]^N$	0
$f_7 = \min\, f_2\left(f_1(x_1, x_2), f_1(x_2, x_3), \ldots, f_1(x_{n-1}, x_n), f_1(x_n, x_1) \right)$	$[-5.12,5.12]^N$	0
$f_8 = \min\, f_2\left(f_1(x_1, x_2) + f_1(x_2, x_3) + \ldots + f_1(x_{n-1}, x_n) + f_1(x_n, x_1) \right)$	$[-5.12,5.12]^N$	0
$f_9 = \min\left(f_2\left(f_1(x_1, x_2) \right) + f_2\left(f_1(x_2, x_3) \right) + \ldots + f_2\left(f_1(x_{n-1}, x_n) \right) \right)$	$[-5.12,5.12]^N$	0

A toolbox for the classical evolutionary algorithm coded in Matlab [11, 12] was employed to investigate the performance of the orthogonal array based crossovers (i.e.: OC, MC and IC), which were embedded in the classical evolutionary algorithm. The objective of solving the benchmark functions is to investigate how better the proposed orthogonal array based crossover (IC) can outperform the other crossovers embedded on the same platform of the classical evolutionary algorithm. We set up

and carried out the experimental work regarding the following settings and configurations that can be classified into two types 1) orthogonal array based evolutionary algorithm embedded with orthogonal array based crossover and 2) standard evolutionary algorithm embedded with standard crossover:

1) The three version of orthogonal array based crossovers (i.e. OC, MC and IC) embedded in the above classical evolutionary algorithm [11, 12] have been tested. They are called **orthogonal array based evolutionary algorithms** in this paper.
    a) The first version is the orthogonal array based evolutionary algorithm (OCEA). The basic process of OCEA is identical to the classical evolutionary algorithm except that the crossover utilizes the orthogonal crossover operator (OC) as discussed in Section 2.1.
    b) The second version is the orthogonal array based evolutionary algorithm (MCEA). The basic process of MCEA is identical to the classical evolutionary algorithm except the crossover utilizes the main effect crossover operator (MC) as discussed in Section 2.2.
    c) The third version is orthogonal array based evolutionary algorithm (ICEA). The basic process of ICEA is identical to the classical evolutionary algorithm except the crossover utilizes the interaction crossover operator (IC) as discussed in Section 3.

An orthogonal array $L_9(3^4)$ [43], which is detailed in Figure A2 in the appendix, has been used in the three orthogonal array based crossover operators (i.e. OC, MC and IC) in all three orthogonal array based evolutionary algorithms (i.e. OCEA, MCEA and ICEA).

2) Two standard evolutionary algorithms (SEAs) have been tested.

a) The first version is the standard evolutionary algorithm one (SEA1). The basic process of SEA1 is identical to that of the classical evolutionary algorithm [11, 12]. The standard three-point crossover is used in SEA1 because three crossover points are produced by the three orthogonal array based crossovers (i.e. OC, MC and IC) with $L_9(3^4)$. To unite the number of crossover points, three crossover points are used in the crossover operator in SEA1.

b) The second version is the standard evolutionary algorithm two (SEA2). The basic process of SEA2 is identical to that of the classical evolutionary algorithm [11, 12] except for the crossover.

In the orthogonal array based crossovers (i.e.: OC, MC and IC), two parents are selected randomly from the population. Then nine potential offspring are produced based on the combinations of the orthogonal array $L_9(3^4)$. In OC, the two resulting children are produced by selecting two best potential offspring from among the nine. In MC, the two children are produced by analyzing the main effects of the genes of the nine offspring. In IC, the two children are produced by analyzing both the main effects of the genes and the interactions between the genes of the nine offspring. Therefore extra selective pressure is created by the three orthogonal array based crossovers (i.e.: OC, MC and IC).

To investigate how the extra selective pressure influences the performance of orthogonal array based evolutionary algorithms, a crossover operator with a parent tournament selection of nine is used in SEA2. In the crossover operator, nine chromosomes are selected randomly from the population. Then the standard three-point crossover is

performed on the two chromosomes with the best fitness among the nine selected chromosomes, and two children are generated for the next generation.

The following parameter values and scheme in the five evolutionary algorithms (i.e.: OCEA, MCEA, ICEA, SEA1 and SEA2) have also been adopted. The dimension of the tested benchmark functions is 30. The pre-defined number of function evaluations in all algorithms is the same, which was set as 200 000. 30 independent runs for each algorithm on each test function have been performed. The real coded representation was used in all algorithms. The parameters of the crossover rate and mutation rate were kept constant and their values were taken from [32]. A mutation rate[1] of 1/30 was used in all algorithms, where 30 is the number of variables in each benchmark function. The mutation operator of Gaussian perturbation of individual variables was used in all algorithms. For the crossover rate, 0.1 was used in the three orthogonal array based evolutionary algorithms (i.e.: OCEA, MCEA and ICEA) and 1.0 was used in the two SEAs (i.e.: SEA1 and SEA2). The value of the crossover rate used in orthogonal array based evolutionary algorithms is smaller than the one used in SEAs, since the orthogonal array based crossover operators (i.e.: OC, MC and IC) are using $L_9(3^4)$ to produce nine potential offspring. With this value of crossover rate, the three orthogonal array based evolutionary algorithms can generate a reasonable number of potential offspring in each generation. A population size of 100 and selective pressure of 1.5 are used in all algorithms, where 1.0 is the minimum selective pressure and 2 is the maximum selective pressure, thus the middle selective pressure 1.5 was used [4, 5].

[1] Muhlenbein [37] recommends the mutation rate of GA as a value of $1/L$, where L is the length of the chromosome.

Table 2 presents the results yielded by the algorithms. The columns show the information in the following order: the name of the benchmark, the means and standard derivations found over the 30 runs by the algorithms SEA1, SEA2, OCEA, MCEA and ICEA. We can see from Table 2 that ICEA outperforms the other algorithms on the nine benchmark functions. ICEA is better than MCEA, which is better than OCEA. Table 1 shows that ICEA achieves the best mean fitness among the five evolutionary algorithms. In fact, ICEA obtains the best mean fitness values in all functions. Also the variances of ICEA are the smallest comparing with the other four evolutionary algorithms in all functions. The smaller the variance, the closer the values cluster around the mean is. Since all the variances of ICEA are the smallest, it is clear that ICEA is capable of approaching and keeping on searching around the mean closer than the other algorithms. Therefore ICEA can produce better quality and more stable solutions than the other four evolutionary algorithms in the nine benchmark functions ($f_1 - f_9$).

T-test was used to evaluate the significance of which ICEA does better than the other algorithms, where the *t*-values are shown in Table 3. It shows that all *t*-values in Table 3 are higher than 2.15. Based on the normal distribution table, if the *t*-value obtained is higher than 1.89, it can be said that the performance of ICEA is better than SEA1, SEA2, OCEA and MCEA with a 97% confidence level in all benchmark functions. Recall that the steps of the algorithms are similar, except that they use different crossover s, where OCEA uses the crossover integrated with orthogonal design and MCEA uses the one considers the main effects in genes. In ICEA, the crossover IC, that considers both main effects in genes and interactions between genes, is used. These results indicate that ICEA outperforms MCEA and OCEA when

the function has high interaction between the variables. These results significantly indicate that IC is an improved crossover operator.

Table 2 Results among the algorithms on $f_1 - f_9$.

	SEA1 Mean (S.D.)	SEA2 Mean (S.D.)	OCEA Mean (S.D.)	MCEA Mean (S.D.)	ICEA Mean (S.D.)
f_1 ($\times 10^1$)	5.5133 (1.8685)	6.0343 (1.9896)	2.8592 (1.34955)	2.8655 (1.3942)	1.6859 (1.2455)
f_2 ($\times 10^{-5}$)	18.0240 (2.6141)	14.0921 (1.8598)	4.6188 (0.9276)	4.5765 (0.6487)	4.2037 (0.6009)
f_3 ($\times 10^1$)	41.3701 (4.5814)	35.0302 (2.8121)	28.0436 (1.6736)	27.9850 (1.3640)	27.2429 (1.1742)
f_4 ($\times 10^{-2}$)	2.3798 (1.8768)	3.7830 (1.8015)	2.4753 (0.7981)	2.3702 (0.8842)	1.9151 (0.2043)
f_5 ($\times 10^0$)	8.0942 (1.9298)	8.2166 (2.0771)	8.0385 (1.6988)	7.7593 (1.4746)	6.6538 (1.3454)
f_6 ($\times 10^{-5}$)	6.2122 (2.7594)	6.2573 (0.9987)	5.0682 (0.7979)	5.0251 (0.8041)	4.6179 (0.3810)
f_7 ($\times 10^0$)	5.8134 (1.9200)	5.4415 (1.8759)	4.8342 (1.3271)	4.1828 (1.1740)	3.5377 (1.0063)
f_8 ($\times 10^0$)	5.3299 (1.7296)	5.5692 (1.7649)	4.5653 (1.7281)	4.0399 (1.6193)	3.8154 (1.4407)
f_9 ($\times 10^0$)	5.9061 (1.8545)	5.2772 (1.8901)	5.0018 (1.8330)	5.0219 (1.8764)	4.5413 (1.6099)

Remarks: The results are averaged over 30 runs. 'Mean' indicates the mean of the best function values found on the evolutionary algorithms. 'S.D.' stands for the standard deviation.

Table 3 T -values between IGA to SEA1, SEA2, OCEA and MCEA

	SEA1 – ICEA	SEA2 – ICEA	OCEA – ICEA	MCEA – ICEA
f_1	9.3355	10.1467	3.4994	3.4560
f_2	38.7302	27.7111	2.0571	2.3092
f_3	16.3608	13.9965	2.1557	2.2584
f_4	4.2495	5.6429	3.7245	2.7468
f_5	3.3536	3.4588	3.4999	3.0334
f_6	3.1348	8.4005	2.7894	2.5066
f_7	5.7500	4.8984	4.2638	2.2851
f_8	4.4267	4.9881	2.4069	1.8928
f_9	6.8146	3.8045	2.6424	2.6649

5 Validation Test

In this section, the case study of the optimization of a car door design [30] was used to validate the effectiveness of ICEA. In the car door design, a fuzzy optimization model was developed which contains the following engineering requirements (i.e. $X=x_1, x_2,\ldots, x_6$) and customer requirements (i.e. $Y=y_1, y_2,\ldots, y_5$):

x_1 – energy to close the door

x_2 – check force on level ground

x_3 – check force on 10% slope

x_4 – door seal resistance

x_5 – Road noise reduction

x_6 – Water resistance

y_1 – easy to close from outside

y_2 – stay open on a hill

y_3 – rain leakage

y_4 – road noise

y_5 – cost

A fuzzy optimization model for the car door design is formulated as shown below:

$$
\begin{aligned}
&\textit{Maximize } \lambda \\
&\textit{subject to} \\
&\lambda \le \mu_{y_i}(X), \qquad i = 1,2,...,5 \\
&\lambda \le \mu_{f_i}(X,Y), \quad i = 1,2,...,5 \\
&\lambda \le \mu_{g_j}(X,Y), \quad j = 1,2,...,6 \\
&\lambda \le \mu_c(X)
\end{aligned}
\tag{14}
$$

where

- $Y=(y_1,y_2,\ldots,y_5)$;
- $X=(x_1,x_2,\ldots,x_6)$;
- $\lambda\,(0 \le \lambda \le 1)$ represents the overall value of membership functions, or overall degree of satisfaction of performance characteristics achieved at a design X;
- membership function $\mu_{y_i}(X)$ can be represented as:

$$
\mu_{y_i}(X) = \begin{cases} 0 & \text{if } y_i(X) \le y_i^{\min} \\ \tau(X) & \text{if } y_i^{\min} \le y_i(X) \le y_i^{\max} \\ 1 & \text{if } y_i(X) \ge y_i^{\max} \end{cases}
\tag{15}
$$

  with the linear or nonlinear fuzzy function $\tau(X)$, and $y_i^{\min}$ and $y_i^{\max}$ represent the lower and upper bounds of aspirations with respect to y_i respectively.

- the membership function of the fuzzy relationship constraints respectively are $\mu_{f_i}(X,Y)$, $\mu_{g_j}(X,Y)$, where $y_i=f_i(x_1,\ldots,x_6)$and $x_j=g_j(x_1,..,x_{j-1},x_{j+1},\ldots,x_6)$ with i=1,2…,5 and j=1,2,…6. The membership functions of a fuzzy constraint "$AX \cong b$"[52] can be represented as:

$$\mu(X)=\begin{cases}0 & if\ AX \le b-d\ or\ AX \ge b+d \\ 1-\dfrac{|AX-b|}{d} & if\ b-d < AX < b+d \\ 1 & if\ AX = b\end{cases} \quad (16)$$

with the row vector *A*, the constant *b* and a chosen constant of admissible violations of the constraint *d*.

- The membership of the cost constraint $\mu_c(X)$ can be represented in the following form:

$$\mu_c(X)=\begin{cases}1 & if\ CX < c \\ 1-\dfrac{CX-c}{t} & if\ c \le CX \le c+t \\ 0 & if\ CX > c+t\end{cases} \quad (17)$$

where *t* is a pre-specified non-negative tolerance level to the cost *c*. By solving the above fuzzy optimization model, an optimal target value setting of the engineering requirements can be obtained. The detailed description of the formulation of the fuzzy optimization model is outside the scope of this paper. For details, the readers can refer to our previous work [3]. This is a non-separable problem since interactions is not avoidable between both engineering requirements (i.e. $X=x_1, x_2,\ldots, x_6$) and customer requirements (i.e. $Y=y_1, y_2,\ldots, y_5$).

The evolutionary algorithms, SEA1. SEA2, OCEA, MCEA and ICEA, were used to solve the optimization problem of determining the target values of the car door design. This is modelled in (18). These algorithms are coded in Matlab. To conduct a more comprehensive comparison, a genetic algorithm was developed which integrated with a gradient search operator proposed by [3] .This algorithm was recoded in Matlab again to solve the problem. We call Bai and Kwong's algorithm BEA in this paper. In these evolutionary algorithms, the population consisted of a set

of real coded chromosomes in which 11 variables are in each chromosome. The *t*-th chromosome in all evolutionary algorithms is represented as:

$$Z(t) = [X(t), Y(t)],$$

where $X(t) = [x_1(t), x_2(t), x_3(t), x_4(t), x_5(t), x_6(t)]$ and $Y(t) = [y_1(t), y_2(t), y_3(t), y_4(t), y_5(t)]$; t=1,2,…,*Popsize* and *Popsize* is the total number of chromosomes in the population. The *t*-th chromosome $Z(t)$ in the evolutionary algorithms is evaluated by the following fitness function the aim of which is to optimize the cost function (18):

$$\begin{aligned}\text{fitness}(Z(t)) &= \text{fitness}(X(t), Y(t)) \\ &= \min\left\{\mu_{y_i}(X(t)), \mu_{f_i}(X(t), Y(t)), \mu_{g_i}(X(t), Y(t)), \mu_c(X(t))\right\}\end{aligned}$$

where i=1,2,…,5; j=1,2,…,6.

The higher the evaluated fitness of the chromosome is, the better the chromosome is. The numbers of evaluations used in the evolutionary algorithms were all the same, set at 80000, which is the same as the one used in [3] for solving the problem. Mutation rate[2] of all evolutionary algorithms was set at 1/11, where there are 11 variable in the chromosomes. The rest of the parameters were set the same as the ones used in Section 4.

Since all evolutionary algorithms are stochastic algorithms, different solutions can be obtained with different runs. The better the evolutionary algorithm is, the larger is the mean and the smaller is the variance of overall customer satisfaction obtained in different runs. Therefore 100 testing runs were performed to collect the two statistics of the means and variances of overall customer satisfaction. These are detailed in Figures 7 and 8 together with the six algorithms. It can be found from Figure 7 that ICEA achieves the largest mean of overall customer satisfaction among the six algorithms and also from Figure 8 that the standard deviation of overall

[2] Muhlenbein [37] recommends the mutation rate of GA as a value of 1/*L*, where *L* is the length of the chromosome.

customer satisfaction with ICEA is the smallest one. Therefore ICEA can yield the best and most robust solutions compared with the other five evolutionary algorithms.

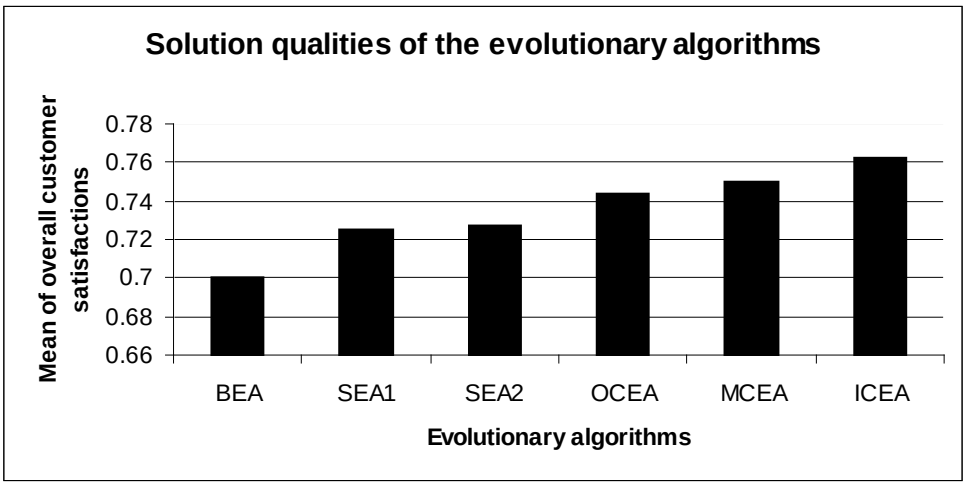

Figure 7 Means of overall customer satisfaction of runs found by the evolutionary algorithms

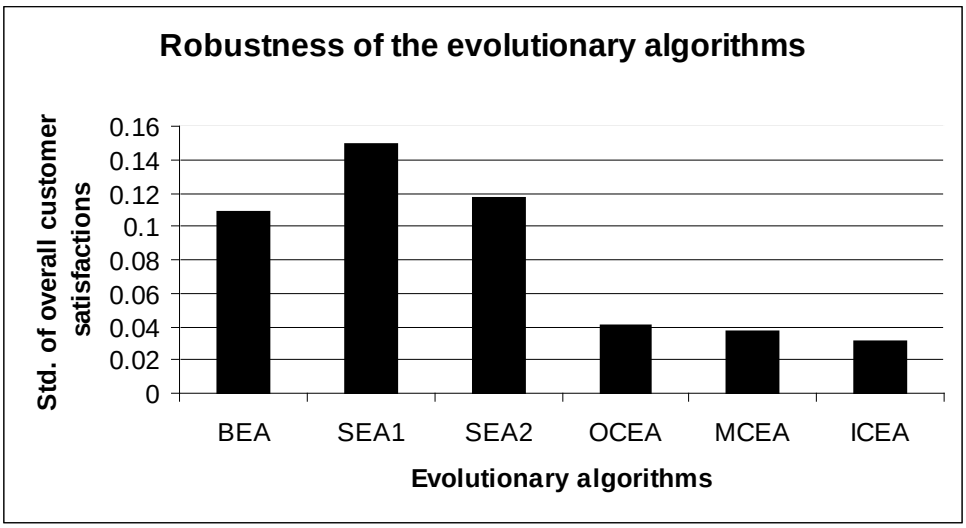

Figure 8 Standard deviations of overall customer satisfaction of runs found by the evolutionary algorithms

The *t*-test is then used to evaluate how significantly better the ICEA is than the other evolutionary algorithms in this validation test. The *t*-values between ICEA and the other evolutionary algorithms are shown in Figure 9, which shows that all *t*-values are higher than 2.15. Based on the normal distribution table, if the *t*-value is higher than 2.15, the significance is the case has a 98% confidence level. Therefore the

performance of ICEA is significantly better than the other five evolutionary algorithms with 98% confidence of solving this problem.

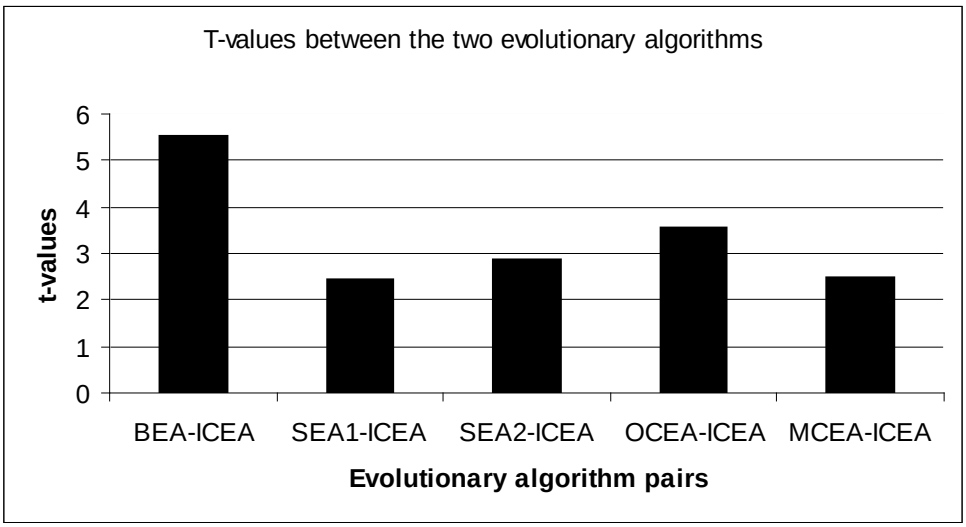

Figure 9 t-values between ICEA to other evolutionary algorithms

After performing the validation test, the convergence plots of all evolutionary algorithms averaged over the 100 runs are shown in Figure 10. The figure shows the progress of the evolutionary algorithms through the searches. It can be observed clearly from the figures that in general the convergence speeds of the orthogonal array based evolutionary algorithms, OCEA, MCEA and ICEA, are in general faster than the other three evolutionary algorithms BEA, SEA1 and SEA2. Finally, it is also obvious that ICEA can produce the better solutions than the other five evolutionary algorithms.

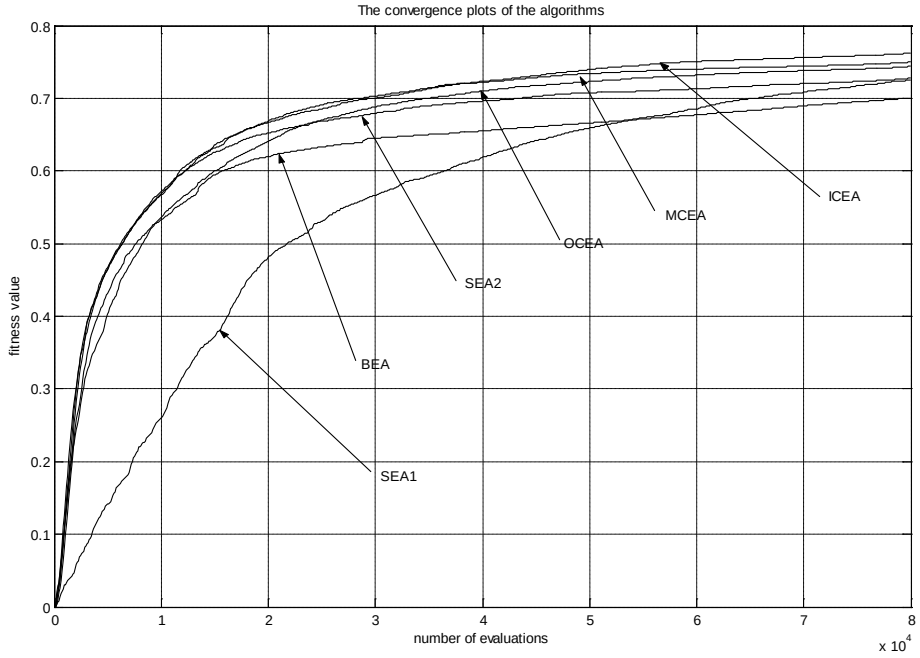

Figure 10 Convergence curves of the evolutionary algorithm for solving the problem of determining the target values in car door design

However, it is hard to count the computational effort used on the algorithms to reach the acceptable solutions, solely from the convergence curves. In [3], it has already been demonstrated that BEA can produce acceptable solutions for this problem. The solutions it found are the most acceptable solutions of this problem. Table 4 shows the computational times used (in seconds) on all algorithms, that can reach the acceptable solutions found by BEA. It can also be found from Table 4 that ICEA can reach acceptable solutions in shortest computational time compared with the other five algorithms. It also shows that ICEA used less than half computational effort to reach the acceptable solutions than BEA [3], even the number of operations used in IC is larger than the other orthogonal array based evolutionary algorithms OCEA and MCEA, and the standard evolutionary algorithms SEA1 and SEA2.

Table 4 Computational time (in seconds) used by the algorithms (i.e. BEA, SEA1, SEA2, OCEA, MCEA and ICEA) until the acceptable solution reached

	BEA	SEA1	SEA2	OCEA	MCEA	ICEA
Computational time taken to reach the acceptable solution	61.2400	44.69	41.61	27.846	23.11	22.21

Recall that the steps of the algorithms are similar, except that different operators are used. In BEA, the gradient search operator is used. In both SEA1 and SEA2, both three point crossovers, one suppressed with normal selective pressure and one suppressed with high selective pressure, are used. In OCEA, OC is used. In MCEA, MC is used. In ICEA, IC is used. These results indicate that IC can aid the evolutionary algorithm to give the best mean solution quality and more robust solutions with the shortest computational time compared with the other algorithms.

7 Conclusion

In this paper, we have proposed a new version of orthogonal array based crossover (IC) that considers the contribution of both the main effect of each individual gene and interactions between genes. It compensates for the potential limitation of the existing versions of orthogonal array based crossovers MC and OC, which ignore interactions between genes. We executed the evolutionary algorithm embedded with the proposed IC, namely ICEA, to solve the nine selected parametrical benchmark problems in which interactions exist between variables. The results show that ICEA can find solutions that are closer to optima than the other evolutionary algorithms embedded with the existing orthogonal array based crossovers (OC and MC) and some other standard crossovers.

To further validate the effectiveness of the proposed IC embedded in the evolutionary algorithm, we applied it to solve the optimization problem of the design of a car door, in which interactions are not avoidable between the engineering requirements and the customer requirements, in the cost function. From the validation test, ICEA was found to yield better results in terms of quality and stability compared with those based on the evolutionary algorithms embedded with the other existing orthogonal array based crossover (OC and MC) and standard crossovers. Referring to the statistical results of the t-test, it can be confirmed that ICEA outperforms significantly the other algorithms involved in the validation test. Also ICEA can reach acceptable solutions with faster convergence speeds and smaller computational effort compared with the other algorithms that were tested.

We are currently implementing the proposed orthogonal array based searching approach embedded in the other stochastic algorithms for solving optimization problems in various product designs. The results will be reported in the future.

Acknowledgement

The authors wish to express their sincere thanks to Colin Reeves and Abdullah Hashim for their very useful comments on the results. They would like to express their gratitude to Angus Wu who suggested they worked on this research topic. They would also like to acknowledge Luis Hercog and James Werner for many useful discussions and valuable suggestions.

Bibliography

[1] G.E.P. Box and W.G. Hunter, J.S. Hunter, *Statistics for Experimenters*. John Wiley, 1978.

[2] T. Back, U. Hammel and H.P. Schwefel, 'Evolutionary computation: comments on the history and current state', *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, April 1997.

[3] H. Bai and C.K. Kwong, 'Inexact genetic algorithm approach to target values setting of engineering requirements in QFD', *International Journal of Production Research*, Vol. 41, No. 16, pp. 3861-3881, 2003.

[4] J.E. Baker, 'Adaptive selection methods for genetic algorithms', in *Proceedings of the First International Conference on Genetic Algorithms*, pp. 101-111, 1985.

[5] J.E. Baker, 'Reducing bias and inefficiency in the selection algorithm', in *Proceedings of the Second International Conference on Genetic Algorithms*, pp. 14-21, 1987.

[6] P.P. Bonissone, R. Subbu, N. Eklund and T.R. Kiehl, 'Evolutionary algorithms + domain knowledge = real-world evolutionary computation', *IEEE Transactions on Evolutionary Computation*, Vol 10, No. 3, pp. 256-280, 2006.

[7] K.Y. Chan, M. Emin Aydin and T.C. Fogarty, 'A Taguchi method-based crossover operator for the parametrical problems', in *Proceedings of the IEEE International Congress on Evolutionary Computation*, pp. 971-977, 2003.

[8] K.Y. Chan, M.E. Aydin and T.C. Fogarty, 'An epistasis measure based on the analysis of variance for the real-coded representation in genetic algorithm', in *Proceedings of the IEEE International Congress on Evolutionary Computation*, pp. 297-304, 2003.

[9] K.Y. Chan, *New Experimental Design Theoretic Genetic Algorithms for Optimisation Problems and Their Application*, MPhil thesis, City University of Hong Kong, March 2003.

[10] K.Y. Chan, *Experimental design techniques in evolutionary algorithms*, PhD thesis, London South Bank University, 2006.

[11] A. J. Chipperfield, P. J. Fleming and C. M. Fonseca, 'Genetic Algorithm Tools for Control Systems Engineering', in *Proceeding of Adaptive Computing in Engineering Design and Control*, pp. 128-133, 1994.

[12] A.J. Chipperfield and P.J. Fleming, 'The MATLAB genetic algorithm toolbox', in *Proceedings of the IEE Colloquium on Applied Control Techniques using MATLAB*, pp. 10/1-10/4, 1995.

[13] D. Cvetkovic and H. Muhlenbein, 'The optimal population size for uniform crossover and truncation selection', in *Technical Report GMD-AS-TR-94-11*, St Augustine, Germany, 1994.

[14] Y. Davidor, 'Epistasis variance: a viewpoint on GA-hardness', In G.J.E. Rawlins (Ed.) *Foundations of Genetic Algorithms*, Morgan Kaufmann, San Mateo, 1991.

[15] C. Dimopoulos and A.M.S. Zalzala, 'Recent developments in evolutionary computation for manufacturing optimization: problems, solutions, and comparisons', *IEEE Transactions on Evolutionary Computation*, Vol. 4, No. 2, pp. 93-113, 2000.

[16] E.J. Davision, *Benchmark problems for control system design*, International Federation of Automatic Control, May, 1990.

[17] D.E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. United States of America: Addison Wesley Longman, Inc, 1989.

[18] S.Y. Ho, L.S. Shu, H.M. Chen, 'Intelligent genetic algorithm with a new intelligent crossover using orthogonal arrays', in *Proceedings of the Genetic and Evolutionary Computation Conference*, vol. 1, pp. 289-296, 1999.

[19] S.Y. Ho, L.S. Shu and J.H. Chen, 'Intelligent evolutionary algorithms for large parameter optimization problems', *IEEE Transactions on Evolutionary Computation*, Vol. 8, No. 6, pp. 522-541, 2004.

[20] S.Y. Ho, H.M. Chen, S.J. Ho, T.K. Chen, 'Design of accurate classifiers with a compact fuzzy-rule base using an evolutionary scatter partition of feature space', *IEEE Transactions on Systems, Man and Cybernetics –Part B: Cybernetics*, Vol. 34, No. 2, pp. 1031-1044, 2004.

[21] S.Y. Ho, J.H. Chen and M.H. Huang, 'Inheritable genetic algorithm for biobjective 0/1 combinatorial optimization problems and it applications', *IEEE Transactions on Systems, Man and Cybernetics –Part B: Cybernetics*, Vol. 34, No. 1, pp. 609-620, 2004.

[22] S.J. Ho, S.Y. Ho, M.H. Hung, L.S. Shu and H.L. Huang, 'Designing structure-specified mixed H_2/H_∞ optimal controllers using an intelligent genetic algorithm IGA', *IEEE Transactions on Control Systems Technology*, Vol. 13, No. 6, pp. 1119-1124, 2005.

[23] S.Y. Ho and H.M. Chen, 'A GA-based systematic reasoning approach for solving traveling salesman problems using an orthogonal array crossover', in *Proceeding of the Fourth International Conference on High Performance Computing in the Asia Pacific Region*, vol. 2, pp. 659-663, 2000.

[24] S.Y. Ho and H.M. Chen, 'An efficient evolutionary algorithm for accurate polygonal approximation', *Pattern Recognition*, Vol. 34, pp. 2305-2317, 2003.

[25], H.L. Huang and S.Y. Ho, 'Mesh optimization for surface approximation using an efficient coarse-to-fine evolutionary algorithm', *Pattern Recognition*, Vol. 36, pp. 1065-1081, 2003.

[26] K. KrishnaKumar, S. Narayanaswamy, and S. Garg, 'Solving large parameter optimization problems using a genetic algorithm with stochastic coding', in *Genetic Algorithms in Engineering and Computer Science*, G.Winter, J. Périaux, M. Galán, and P. Cuesta, Eds. New York:Wiley, 1995.

[27] C.K. Kwong, K.Y. Chan, M.E. Aydin and T.C. Fogarty, 'An orthogonal array based genetic algorithm for developing neural network based process models of fluid dispensing', *International Journal of Production Research*, Vol. 44, No. 12, pp. 4815-4836, 2006.

[28] A.I. Khuri and J.A. Cornell, *Response Surfaces Design and Analysis*. New York: Marcel Dekker, Inc, 1996.

[29] J.D. Kim and M.S. Choi, Stochastic approach to experimental analysis of cylindrical lapping process, *International Journal of Machines Tools Manufacturing*, Vol. 35, No. 1, pp. 51-59, 1995.

[30] K. Kim, H. Moskowitz, A. Dhingra and G. Evans, 'Fuzzy multicriteria models for quality function deployment', *European Journal of Operational Research*, 121, 504-518, 2000.

[31] Y.W. Leung and Y. Wang, 'Multiobjective programming using uniform design and genetic algorithm', *IEEE Transactions on Systems, Man, and Cybernetics – Part C: Applications and Reviews*, vol. 30, no. 3, pp. 293-304, 2000.

[32] Y.W. Leung and Y. Wang, 'An orthogonal genetic algorithm with quantization for global numerical optimization', *IEEE Transactions on Evolutionary Computation*, vol. 5, No. 1, pp. 41-53, 2001.

[33] Y.H. Lin, Y.Y. Tyan and T.P. Chang and C.Y. Chang, 'An assessment of optimal mixture for concrete made with recycled concrete aggregates', *Cement and Concrete Research*, Vol. 34, pp. 1373-1380, 2004.

[34] K.F. Man, K.S. Tang and S. Kwong, 'Genetic algorithms: concepts and applications', *IEEE Transactions on Industrial Electronics*, vol. 43, no. 5, 1996.

[35] N.S. Mohan, A. Ramachandra and S.M. Kulkarni, 'Influence of process parameters on cutting force and torque during drilling of glass fiber polyester reinforced composites', *Composite Structures*, Vol. 71, pp. 407-413, 2005.

[36] D.C. Montgomery, *Design and Analysis of Experiments*, New York: John Wiley and Sons, Inc, 1997.

[37] H. Muhlenbein, 'How genetic algorithms really work-Part I: Mutation and hillclimbing', in *Proceedings of the 2nd International Conference on Parallel Problem Solving from Nature*, pp. 15-25, 1992.

[38] M.S. Phadke, *Quality engineering using robust design*, New York: Prentic Hall, 1987.

[39] C.R. Reeves and C.C. Wright, 'An experimental design perspective on genetic algorithms', in *Foundation of Genetic Algorithms 3*, pp. 7-22, 1995.

[40] C.R. Reeves and C.C. Wright, 'Epistasis in Genetic Algorithms: An Experimental Design Perspective', in *Proceedings of the 6th International Conference on Genetic Algorithms*, pp. 217-224, 1995.

[41] C.R. Reeves, 'Predictive measures for problem difficulty', in *Proceedings of the 1999 Congress on Evolutionary Computation*, vol. 1, pp. 736-742, 1999.

[42] D.I. Seo and B.R. Moon, 'A survey on chromosomal structures and operators', in *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 1357-1368, 2003.

[43] G. Taguchi and S. Konishi, *Orthogonal Arrays and Linear Graphs*. Dearbon, MI: American Supplier Institue, 1987.

[44] D. Thierens, D. E. Goldberg, and A. G. Pereira, 'Domino convergence, drift, and the temporal-salience structure of problems', in *Proceedings of IEEE International Conference of Evolutionary Computation*, pp. 535-540, 1998.

[45] 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.

[46] R. Unal, D.O. Stanley and C.R. Joyner, 'Propulsion system design optimization using the Taguchi Method', *IEEE Transactions on Engineering Management*, vol. 40, no. 3, pp. 315-322, August 1993.

[47] D. Whitley, 'The genitor algorithm and selective pressure: why rank-based allocation of reproductive trials is best', in *Proceedings of the Third International Conference on Genetic Algorithms*, pp. 116-121, 1989.

[48] D. Whitley, K. Mathias, S. Rana and J. Dzubera (1995), 'Building better test function', in *Proceedings of the 6th International Conference on Genetic Algorithms*, pp. 239-246, 1995.

[49] X. Yao, Y. Lin and G. Lin, 'Evolutionary programming made faster', *IEEE Transactions on Evolutionary Computation*, Vol. 3, No. 2, pp. 82-102, 1999.

[50] G.Z. Yin and D.W. Jillie, 'Orthogonal design for process optimization and its application in plasma etching', *Taguchi Methods: Applications in World Industry*, A. Bendell, J. Disney, W.A. Pridmore (ed.s), IFS Publications/Springer-Verlag, pp. 181-198, 1989.

[51] Q. Zhang and Y.W. Leung, 'An orthogonal genetic algorithm for multimedia multicast routing', *IEEE Transactions on Evolutionary Computation*, Vol. 3, No. 1, pp. 53-62, 1999.

[52] H.J. Zimmermann, *Fuzzy Set Theory and Its Applications*, 3rd Edition Boston: Kluwer, 1996.

Appendix:

Run	*1st parameter*	*2nd parameter*	*3rd parameter*
1	1	1	1
2	1	2	2
3	2	1	2
4	2	2	1

Figure A1: The orthogonal array $L_4(2^3)$

Run	*1st parameter*	*2nd parameter*	*3rd parameter*	*4th parameter*
1	1	1	1	1
2	1	2	2	2
3	1	3	3	3
4	2	1	2	3
5	2	2	3	1
6	2	3	1	2
7	3	1	3	2
8	3	2	1	3
9	3	3	2	1

Figure A2: The orthogonal array $L_9(3^4)$

Example A1

'arg' is a function that returns the indices of the minimum value of the matrix. This function can return the positions of the element with minimum value in each column. It can also return the position of the element with minimum value in the matrix.

For example,

$$[3,1,2,1] = \arg\left(\min_{j=1,2,3,4} \begin{pmatrix} 5 & 1 & 2 & 1 \\ 4 & 5 & 0 & 2 \\ 3 & 2 & 3 & 3 \end{pmatrix} \right)$$

The third element in the first column is the minimum value among the elements in the first column. The first, second, and first elements in the second, third and fourth columns are the minimum one among the elements in the correspondent column. Therefore the result of this function is [3,1,2,1].

For another example,

$$[3,2] = \arg\left(\min_{i=1,2,3;\, j=1,2,3,4} \begin{pmatrix} 5 & 1 & 2 & 1 \\ 4 & 5 & 0 & 2 \\ 3 & 2 & 3 & 3 \end{pmatrix} \right)$$

The value '0' is the minimum value among all elements of the matrix. It is located in the third column and the second row of the matrix. Therefore the result of this function is [3,2].