Improved orthogonal array based simulated annealing for design optimization

Chan, K.Y. Kwong, C.K.* and Tsim, Y.C.

Department of Industrial and Systems Engineering,

The Hong Kong Polytechnic University, Hung Hom, Kowloon

Hong Kong, PRC

Abstract

Recent research shows that simulated annealing with orthogonal array based neighbourhood functions can help in the search for a solution to a parametrical problem which is closer to an optimum when compared with conventional simulated annealing. Previous studies of simulated annealing analyzed only the main effects of variables of parametrical problems. In fact, both main effects of variables and interactions between variables should be considered, since interactions between variables exist in many parametrical problems. In this paper, an improved orthogonal array based neighbourhood function (IONF) for simulated annealing with the consideration of interaction effects between variables is described. After solving a set of parametrical benchmark function problems where interaction effects between variables exist, results of the benchmark tests show that the proposed simulated annealing algorithm with the IONF outperforms significantly both the simulated annealing algorithms with the existing orthogonal array based neighbourhood functions and the standard neighbourhood functions. Finally, the

improved orthogonal array based simulated annealing was applied on the optimization of emulsified dynamite packing-machine design by which the applicability of the algorithm in real world problems can be evaluated and its effectiveness can be further validated.

Keywords: Orthogonal array, interaction analysis, neighbourhood functions, simulated annealing, design optimization

Corresponding author:

Dr. C.K. Kwong

Department of Industrial and Systems Engineering, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong, PRC

Tel: (852) 27666610

Fax: (852) 23625267

Email: mfckkong@polyu.edu.hk

1. Introduction

Simulated annealing (SA) is a point-based stochastic optimization method, which explores iterationally from an initial solution to the optimum [4, 14]. Each iteration employs a neighbourhood function to generate a candidate solution by a randomized perturbation on a current solution. Therefore, design of neighbourhood functions plays an important role in developing an effective simulated annealing. The searching mechanism of SA has a very good convergent property [19] and SA has been widely applied in solving many hard optimization problems [32, 33]. However, it can be noted in many previous researches [1, 17, 26] that SA can find good or reasonable solutions, but in many cases it cannot search for a global optimum. The searching ability of SA improves in the earlier stages of the searching process, but it saturates or even terminates in later stages. Therefore, it is difficult to obtain any substantial improvements by examining neighbouring solutions in the later stage of the search. Vaessens et al. [34] put this searching method into the context of a local, or neighbourhood search. Also it has been noted in some previous researches that long computational time is commonly required for SA to search for an acceptable solution for solving hard optimization problems [29, 38, 39]. Various approaches have been proposed to improve the searching mechanism by modifying neighbourhood functions [8, 38, 39, 30], modifying criterion of accepting a new candidate solution [28], incorporating with other optimization methods [7, 21, 36] and parallelized computing [1].

A recent approach to improving the searching mechanism of SA has been proposed by introducing orthogonal arrays into neighbourhood functions of SA [9, 10, 12, 27]. The Orthogonal arrays exploit the neighbourhood of a current solution by

analyzing the main effect of variables in the current solution. The neighbourhood function, which uses the orthogonal array for exploiting solution spaces, is called orthogonal array based neighbourhood function (ONF) in this paper. It has been shown that ONF can speed up the search of SA and determine a more accurate candidate solution for electromagnetic problems [27], floorplanning problems [10] and controller design problems [9, 12] compared with other neighbourhood functions. However, we found that the exploitation of candidate solutions can be further improved by considering not only the main effects of variables but also the interaction effects between variables. It is due to the fact that strong interaction effects could exist between variables in many optimization problems. If strong interaction effects exist in localized features of a search space, poor results may be obtained by considering main effects only in variables [6, 23, 24, 25]. In this paper, an improved orthogonal array based neighbourhood function (IONF) for SA which considers both main effects in variables and interaction effects between variables is proposed. Application of the proposed simulated annealing algorithm with the IONF on the optimization of emulsified dynamite packing-machine design is also described. IONF employs the approach of interaction plots [22] to analyze interaction effects between variables. Interaction plots have been commonly used to analyze interaction effects between parameters in industrial systems [31, 13, 18, 20, 40].

The background of orthogonal arrays and ONF, as well as the proposed IONF are described in Section 2 and 3 respectively. Benchmark results of solving a set of hard benchmark problems [37, 11] using the three simulated annealing algorithms with employing ONF, IONF and the standard neighbourhood function respectively are shown in Section 4. Application of the improved simulated annealing on the optimization of

emulsified dynamite packing-machine design and further validation of the effectiveness of the algorithm are described in Section 5.

2. Orthogonal Experimental Design and Neighbourhood Function

The use of orthogonal arrays in planning experiments and analyzing experimental data is briefly described in Section 2.1. In Section 2.2, background and limitations of the orthogonal array based neighbourhood functions (ONF), that has been applied in solving many hard optimization problems [9, 10, 12 and 27], are presented.

2.1 Orthogonal Experimental Design

One major objective of an experimental design is to find the best combination of parameter levels for optimal performance of a system or a model. If an experimental design is based on the full factorial one and the number of parameters to be investigated is large, a large number of experimental runs always are required to be carried out. To reduce the number of experimental runs, a fractional factorial design is an alternative in which the experimental design can be based on orthogonal arrays [2]. An experimental plan based on an orthogonal array $L_{2N+1}(p^N)$ involves a maximum N parameters and p levels in each parameter for $2N+1$ experimental runs.

If an orthogonal array $L_{2N+1}(3^N)$ with 3 levels is considered, for j=1,2,…N and k = 1,2,3, main effect M_{jk} of the parameter j for level k is defined as:

$$M_{jk} = \sum_{t=1}^{2N+1} y_t \times F_t \qquad (1)$$

where y_t denotes an objective function value of the combination corresponding to experiment t, and

$$F_t = \begin{cases} 1 \text{ if level of parameter } j \text{ of experiment } t \text{ is } k. \\ 0 \text{ if level of parameter } j \text{ of experiment } t \text{ is not } k. \end{cases}$$

For smaller-the-better type problems, the best level *Best*(*j*) of the *j*-th parameter is denoted as:

$$Best(j) = \arg\left(\min\left(M_{j1}, M_{j2}, M_{j3}\right)\right) \quad (2)$$

where 'arg(min(..))' is a function that returns the index of the minimum value. For example, if the value of M_{j2} is the smallest among the values of M_{jk} where k=1,2 and 3, then *Best*(*j*)=2. For larger-the-better type problems, the best level *Best(j)* of the *j*-th parameter is denoted as:

$$Best(j) = \arg\left(\max\left(M_{j1}, M_{j2}, M_{j3}\right)\right) \quad (3)$$

where ' arg(max(..))' is a function that returns the index of the maximum value.

2.2 Orthogonal Array based Neighbourhood Function (ONF)

Ho et al [9, 10, 12] and Shu *et al* [27] proposed an orthogonal array based neighbourhood function (ONF) which aims at generating a candidate solution by using the combinations of the orthogonal array $L_{2N+1}\left(3^N\right)$. ONF generate a candidate solution Q=ONF(P_1) from P_1, where $P_1 = \left(S_1, ..., S_m\right)$ is the current solution. First two temporary solutions, P_2 and P_3 are generated by perturbing P_1 as follows:

$$P_2 = \left(S_1^1, ..., S_m^1\right) \text{ and } P_3 = \left(S_1^2, ..., S_m^2\right), \quad (4)$$

where $S_i^1 = S_i + \overline{S_i}$ and $S_i^2 = S_i - \overline{S_i}$, i=1,...,m. All $\overline{S_i}$ are generated based on the Cauchy-Lorentz probability distribution [29]. Consequently, Q is produced by a combination of variables of P_1, P_2, and P_3.

In ONF, P_1, P_2 and P_3 are considered as level 2, level 1 and level 3 of an experimental design. To assign the m variables from P_1, P_2 and P_3 into the N parameters in $L_{2N+1}\left(3^N\right)$, the m variables are divided into N non-overlapping groups. Each non-overlapping group of variables is considered as a parameter in $L_{2N+1}\left(3^N\right)$. The number of variables in each group is determined by a generated random integer l_i, i=1,2,…N, where

$$\sum_{i=1}^{N} l_i = m \tag{5}$$

ONF then uses the t-th combination of $L_{2N+1}\left(3^N\right)$ to compute y_t corresponding to the t-th experiment with t=1,...2N+1, and computes all main effects M_{jk} with j=1,2,..,N and k=1,2,3 based on (1). Finally, it determines the best level of each parameter based on the computed main effects by (2) for smaller-the-better type problems or (3) for larger-the-better type problems. The algorithm of the ONF, Q=ONF(P_1), is shown in the Appendix 1.

ONF uses the analysis of main effects to determine the optimal levels of parameters which is the simplest approach to analyze experimental results [2, 22]. However, it is quite common that an interaction effect exists between two parameters in a function [6]. Further studies of interaction effects and main effects in a function have been done by [23, 24, 25] based on 'analysis of variance (ANOVA)'. The parametrical effects of a function are analyzed using a total sum of squared deviations SS, which can be divided into SS of main effects and interaction effects as shown below:

Total SS = SS of main effects + SS of interactions

With the higher SS of interactions, the lack of provision of adequacy dealing with the potential interactions between parameters is a major weakness of ONF. To solve the

optimization problems where low interaction effects exist between parameters, ONF could work properly. However, if strong interaction effects exist between parameters in optimization problems, the optimal combination based on ONF may not be reproducible.

3 An Improved Orthogonal Array Based Neighbourhood Function

In this paper, an improved orthogonal array based neighbourhood function (IONF) is proposed with the consideration of interaction effects between parameters. In the ONF, the children are produced by considering only the largest main effects of parameters. But, in IONF, the children are produced by considering both main effects of parameters and interaction effects between parameters. Interaction plots [22] are adopted to investigate the magnitudes of interaction effects between parameters.

In the IONF, an interaction matrix MI_{ij} is generated to estimate the magnitudes of interaction effects between parameters i and j, where i, j=1,2,..N. It can be expressed as:

$$MI_{ij} = \left(I_{ij}(m,n); \text{for } 1 \le \text{m, n} \le 3\right)_{3\times 3} \tag{6}$$

where the numbers of rows and columns of the interaction matrix MI_{ij} are both equal to the number of levels of $L_{2N+1}(3^N)$ which is 3. The elements of MI_{ij}, $I_{ij}(m,n)$, which represents the average fitness of the i^{th} parameter with level m and j^{th} parameter with level n, are defined as:

$$I_{ij}(m,n) = \frac{\sum_{p=1}^{N} f_p \cdot \left[\begin{array}{l}\text{In the } p^{th} \text{ combination of } L_{2N+1}\left(3^N\right), \text{the level of the } i^{th} \text{ parameter is } m \\ \text{and the } j^{th} \text{ parameter is } n\end{array}\right]}{\sum_{p=1}^{N} \left[\begin{array}{l}\text{In the } p^{th} \text{ combination of } L_{2N+1}\left(3^N\right), \text{the level of the } i^{th} \text{ parameter is } m \\ \text{and the } j^{th} \text{ parameter is } n\end{array}\right]} \tag{7}$$

where f_p denotes an objective function value of the combination corresponding to the p-th row of $L_{2N+1}(3^N)$, $1 \le m, n \le 3$ and

$$\left[\text{condition}\right] = \begin{cases} 1 \text{ if the statement inside the bracket is true.} \\ 0 \text{ otherwise.} \end{cases}$$

Then interaction plots are used to investigate the magnitude of interaction effects between parameters i and j. The r^{th} line of the interaction plot is defined as:

$$Line_{ij}(r) = \left(I_{ij}(1,r), I_{ij}(2,r), I_{ij}(3,r)\right) \text{where } r = 1, 2, 3. \quad (8)$$

Figure 1 shows that the lines cross indicating the existence of strong interactions. If strong interaction effects do not exist in all parameter pairs, only the main effects of parameters need to be studied. The candidate solution Q of the IONF is generated by the combination of the parameters with the largest main effects based on (2) for smaller-the-better type problems or (3) for larger-the-better type problems. In this case, the algorithm of IONF is identical to the one of ONF. However, if strong interaction effects exist in any one of the parameter pairs, the candidate solution Q is first generated by the best level combinations of the orthogonal array $L_{2N+1}(3^N)$ with the optimal y_t. For those parameters without strong interaction effects between each other, the level combinations in Q are replaced by the parameters with the largest main effects based on (2) or (3).

The algorithm of the IONF, Q=IONF(P_1), is given as follows:

Algorithm Q=IONF(P_1)

Step 1) Generate P_2 and P_3 with $P_1 = \left(S_1, ..., S_m\right)$ based on (4).

Step 2) Divide P_1, P_2 and P_3 into N groups based on (5).

Step 3) Represent levels 2, 1 and 3 of the j-th parameter of $L_{2N+1}\left(3^N\right)$ by the j-th group of P_1, P_2 and P_3 respectively.

Step 4: Compute y_t based on the t-th combination of $L_{2N+1}\left(3^N\right)$ of the t-th experiment.

Step 5: Compute the main effects M_{jk} where j=1,...,N and k=1,2 and 3 based on (1).

Step 6: Construct the interaction matrices MI_{ij} by (6), where i, j=1,2,…N with $i \neq j$.

Step 7: Construct the interaction plot for MI_{ij} where i, j=1,2,…N with $i \neq j$.

Step 8: Check whether the parameters i and j have a strong interaction effect of each other, where i, j=1,2,…N with $i \neq j$.

Step 9: If strong interaction effect exists in any one of the parameter pairs, goto Step 10, otherwise goto Step 13.

Step 10: Form the candidate solution Q by the combination of the $L_{2N+1}\left(3^N\right)$ with the optimal y_t.

Step 11: For the parameter pair with no strong interaction effect, the level combinations in Q are replaced by the level combinations with the largest main effects based on (2) or (3).

Step 12: Output Q as the resulting solution of IONF. Then goto step 15.

Step 13: Determine the best level *Best*(j) on the j-th parameter based on (2) for smaller-the-better type problems and (3) for larger-the-better type problems.

Step 14: The candidate solution Q is produced by the combinations of best levels of parameters.

Step 15: Terminate the algorithm.

4 Benchmark results based on non-separable functions

To evaluate the effectiveness of the proposed IONF, benchmark tests based on the three simulated annealing algorithms, which employ the standard neighbourhood function SNF [32], ONF [9, 10, 12, 27], and the proposed IONF respectively, were conducted to solve a set of selected parametrical benchmark functions. Those benchmark functions ($f_1 - f_6$) is shown in Table 1, in which interaction effects exist between variables. $f_1 - f_4$ were collected from [12], while $f_5 - f_6$ were collected from [37]. They cannot be decomposed into linear combinations of independent sub-functions since variables interact with each other and cannot be enumerated completely. They can be classified as good test suites for the algorithms since they are non-separable in which each sub-function contains at least two variables [35].

To evaluate the performance of the three neighbourhood functions (SNF, ONF and IONF), the simulated annealing algorithm used by [12, 27] was employed in this research which is shown below.

Algorithm of simulated annealing

Begin
i=1
Randomly generate a candidate solution s
Repeat
$t=T_0$; $I=I_0$
Repeat:-
Q = neighbour_function(s)
if f(Q)< *f*(s) *then replace s with Q*
else
generate a random number r
if exp(-(*f*(*s*)-*f*(*Q*))/*t*)>*r then replace s with Q*
endif
endif
*t=t*CR*
*until t<I*CR*
i=i+1
Until i=(pre-defined number of function evaluations)
End.

In the algorithm, a candidate solution *s* is randomly initialized first, and then it starts with the highest temperature ($t=T_0$). The algorithm then modifies the candidate solution *Q* by using a neighbourhood function and judges if the yielded solution will be promoted for the next inner iteration. Then the algorithm reduces the temperature *t* to *t=t*CR* by a cooling coefficient (*CR*) at the end of each inner iteration. When the temperature reduces to *I*CR*, the solution *Q* results from the inner iteration. The simulated annealing algorithm stops iterating until *i* reaches the pre-defined number of functional evaluations.

The three simulated annealing algorithms with SNF, ONF and IONF respectively are called standard simulated annealing (SSA), orthogonal simulated annealing (OSA) and improved orthogonal simulated annealing (IOSA). They have been implemented to solve the benchmark functions f_1 to f_6. By referring to the previous research [9], the parameters used in OSA and IOSA are T_0=50, I_0=5 and *CR*=0.95. The orthogonal array

used in OSA and IOSA is also same as the one used in [9], which is $L_{2N+1}(3^N)$, where $N = \left(3^{\lfloor \log_3(2p+1) \rfloor} - 1\right)/2$ and p is the number of variables of the optimization problem.

Since only one functional evaluation is implemented in SNF, a functional evaluation is required in each iteration of SSA. However, $2N+1$ functional evaluations are implemented in the ONF and IONF. Therefore, $2N+1$ functional evaluations are required in each iteration of the OSA and IOSA. To make the number of functional evaluations of all the algorithms to be the same, the number of iterations used in SSA is larger than those used in both the OSA and IOSA. Therefore, value of *CR* used in SSA is larger than the one used in the OSA and IOSA. The parameters of SSA used are T_0=50, I_0=5 and *CR*=0.99. The pre-defined number of functional evaluations used in all the algorithms is 10 000.

Tables 2 to 7 as shown in Appendix 2 are the statistical results based on the algorithms for solving the functions f_1 to f_6 respectively, where the dimensions, p= 20, p= 40, p= 60, p= 80 and p= 100, are used. The statistical results of means and standard deviations of the solutions over the 30 runs are shown as well. It can be seen from the tables that the mean values based on the IOSA are smaller than those based on the OSA or SSA. Also the standard deviations of IOSA are found to be the smallest compared with those of the other two simulated annealing algorithms in all benchmark functions. Therefore, in terms of quality and robustness of the solutions obtained, IOSA outperforms the other two simulated annealing algorithms in all the six benchmark functions. Based on Tables 2 to 7, Figures 2 to 7 can be generated, which show the means of the solutions based on the three simulated annealing algorithms respectively for the six benchmark functions. It can be found from the figures that the IOSA outperforms the

OSA and the SSA in all the six benchmark functions with various number of dimensions. This indicates that the IOSA can perform well in solving the problems where interaction effects exist between variables with small and large numbers of dimensions.

5 Application of IOSA on design optimization and further validation

In weapon manufacturing, handling of dynamite is in powder state which is easy to explode during transportation or storage. Nowadays, dynamite is first emulsified into liquid state which is more safe in handling. A machine normally is required to perform the emulsification of dynamite which is commonly named as an emulsified dynamite packing machine. In this section, optimization of emulsified dynamite packing machine design based on the IOSA is illustrated. An optimization problem of the machine design formulated by [16] was adopted in this research which contains the following engineering requirements (i.e. X=X_1, X_2,…, X_7) and customer requirements (i.e. Y=Y_1, Y_2,…, Y_4):

X_1 – precision of molding of clip

X_2 – precision of dynamite packing

X_3 – control force of dynamite packing

X_4 – efficiency of dynamite packing

X_5 – hardness of pressing hammer

X_6 – noise of cam power transmission

X_7 – height of machine bed

Y_1 –quality of packing dynamite

Y_2 – efficiency of packing dynamite

Y_3 – packing noise

Y_4 – rigidity of machine

Details of the optimization problem [16] are shown below.

$$osc = \left(0.46y_1^r + 0.28y_2^r + 0.16y_3^r + 0.10y_4^r\right)^{\frac{1}{r}} \tag{9}$$

subject to:

$$y_1 = 5.98x_1 - 0.33x_2 - 1.37x_3 + 0.88 \quad (10)$$

$$y_2 = 2.45x_3 + 0.96x_4 + 1.25x_5 + 0.54 \quad (11)$$

$$y_3 = 4.20x_6 + 1.00 \quad (12)$$

$$y_4 = 4.00x_7 + 1.25 \quad (13)$$

$$(f_1 - 1.3)^2 + (f_2 - 1)^2 \le 0.5^2 \quad (14)$$

$$f_1 = 0.464y_1 + 0.449y_2 - 0.217y_3 + 0.166y_4 \quad (15)$$

$$f_2 = -0.030y_1 - 0.100y_2 - 0.508y_3 - 0.695y_4 \quad (16)$$

$$50 + 20x_1 + 25x_2 + 10x_3 + 15x_4 + 5x_5 + 30x_6 + 8x_7 \le 100 \quad (17)$$

$$1 \le y_i \le 5, i = 1,...,4 \quad (18)$$

$$0 \le x_j \le 1, \ j = 1,...,7 \quad (19)$$

where

- x_i (i=1,2,…7) is the level of attainment of X_i; y_i (i=1,2,…4) is the value of customer satisfaction of Y_i;
- (9) is the objective function of deriving overall customer satisfaction (OCS);
- (10) to (13) are the models of functional relationship between customer requirement Y_i, i=1,…,4 and engineering requirements, X_j, j=1,…,7;
- (14) to (16) are the constraints of product positioning;
- (17) is the cost constraint that is subject to a budget with the fixed cost and the cost incurred for achieving each engineering requirement, X_j, j=1,…,7;
- (18) and (19) are the ranges of values of the customer satisfactions and levels of attainment of the engineering requirements respectively.

The IOSA was used to determine the optimal setting of levels of attainment of the engineering requirements, x_1, x_2, x_3, x_4, x_5, x_6 and x_7, by maximizing the overall customer satisfaction. The algorithm was implemented by using Matlab, in which a candidate solution is represented as:

$$s = [x_1, x_2, x_3, x_4, x_5, x_6, x_7] \tag{20}$$

The objective function used in the algorithm is defined by maximizing the optimization function (9) subject to the constraints. It is defined as:

$$f(s) = \max\left(\begin{cases} \min(-f_b(s), -f_c(s)) & \text{if } f_b(s) < 0 \text{ or } f_c(s) < 0 \\ f_a(s) & \text{otherwise} \end{cases} \right) \tag{21}$$

where $f_a(s) = \left(0.46 y_1^r + 0.28 y_2^r + 0.16 y_3^r + 0.10 y_4^r\right)^{\frac{1}{r}}$; (22)

$$f_b(s) = 0.5^2 - \left[(f_1 - 1.3)^2 + (f_2 - 1)^2\right]; \tag{23}$$

$$f_c(s) = 100 - (50 - 20x_1 - 25x_2 - 10x_3 - 15x_4 - 5x_5 - 30x_6 - 8x_7); \tag{24}$$

$$f_d(s) = \sum_{i=1}^{4} (y_i - 3)^2 \cdot \mathrm{R}_i; \mathrm{R}_i = \begin{cases} 1 \text{ if } y_i < 1 \text{ and } 5 < y_i \\ 0 \text{ if } 1 \le y_i \le 5 \end{cases} \tag{25}$$

y_1, y_2, y_3 and y_4 can be found in (10), (11), (12) and (13) respectively; f_1 and f_2 can be found in (15) and (16) respectively. (23) is formulated to handle the constraints (14)-(16). (24) and (25) are formulated to handle the constraints (17) and (18) respectively. The constraint (19) can be dealt with by setting the ranges of solutions in the simulated annealing.

The parameter setting used in the IOSA is the same as the one used in Section 4 except the number of functional evaluations. The variable *r* used in (25) was set to 1, 2, 3,

4 and 5. With a higher value of *r*, interaction effects between variables are higher. Since the simulated annealing algorithm is stochastic one, different solutions are obtained from different runs. Therefore, 30 test runs were performed in order to obtain the two statistics, means and variances of overall customer satisfaction with respect to the values of *r*. After assessing the interaction effects between the requirements as shown in the house of quality for the machine design [16], they were assessed as 'low to medium' and hence *r* value was set as 2. After running the IOSA for 30 times, the optimal solutions *s* corresponding to *r*=2 was found to be (0.8735, 0.0002, 0.8055, 0.5479, 0.9913, 0.3211, 0.2076).

To further evaluate the effectiveness of the IOSA in solving real world optimization problems compared with other simulated annealing algorithms, SSA and OSA were also used to solve the same optimization problem using the same parameter setting and number of test runs. Table 8 shows the means and standard deviations of the runs with different values of *r*. It can be found from the table that the means of the overall customer satisfaction based on the IOSA for any value of *r* are larger than the ones based on the OSA and SSA, except *r*=1. It is because there is no interaction between variables of (9) while *r*=1. Therefore the overall mean of customer satisfaction based on the IOSA is nearly identical to that based on the OSA. Regarding the standard deviation, it can be found from Table 8 that the average standard deviations of the IOSA is smaller than the ones based on the OSA and SSA.

The *t*-test was conducted to evaluate how significantly the IOSA is better than the other two simulated annealing algorithms in this optimization problem. The *t*-values between the IOSA and the other two simulated annealing algorithms are shown in Figure

8, from which it can be seen that all the *t*-values are higher than 1.675 except the *t*-value for OSA-IOSA while *r*=1. Based on the normal distribution table, if the *t*-value is higher than 1.675, the difference of performance between two algorithms is significant with a confidence level of 95.3%. Therefore it can be concluded that the performance of the IOSA is significantly better than the SSA and OSA. As explained before, while *r*=1, interaction effects does not exist in (9). There is no significant difference in performance between the OSA and IOSA. The significance of the difference increases as *r* increases. It can be explained that with a larger value of *r*, interaction effects between variables become stronger.

On the other hand, computational times of generating solutions based on the IOSA were also compared with those based on the OSA and SSA. Figure 9 shows the computational times of the three algorithms with respect to different values of *r*. Execution of the algorithms is based on a Pentium 4 PC with 2.26 MHz. It can be seen from the figure that the times taken to search for solutions based on the IOSA are less than those based on the OSA and SSA for all values of *r*, except *r*=1. When *r*=1, the time taken based on the IOSA is identical to that based on the OSA, but is still less than that based on the SSA.

6 Conclusion and further work

In this paper, an improved orthogonal array based neighbourhood function (IONF) for simulated annealing which considers both main effects of individual variable and interaction effects between variables is described. The proposed IONF is to make up the deficiency of the existing orthogonal array based neighbourhood functions (ONF), which

is lack of consideration of interaction effects between variables. Benchmark tests involving 6 parametrical benchmark functions were conducted based on the 3 simulated annealing algorithms, IOSA, OSA and SSA, which were incorporated with the IONF, ONF and SNF respectively. Results of the benchmark tests indicate that IOSA outperforms OSA and SSA in terms of quality and robustness of solutions.

The IOSA was successfully applied to solve the design optimization problem of emulsified dynamite packing machines in which interaction effects exist between the requirements. Further validation tests based on the case of the machine design were conducted in order to evaluate the effectiveness of the IOSA in solving real world optimization problems compared with the OSA and SSA. Results of the validation tests also indicate that IOSA outperforms the other two simulated annealing algorithms in terms of quality and robustness of solutions. T-tests were also conducted. Results of the tests indicate that the IOSA outperforms the other two algorithms significantly. Besides, the times taken to search for solutions based on the IOSA are the smallest compared with those based on the other two algorithms.

Further work would involve the study of interaction effects among three or more variables in the IOSA. On the other hand, the orthogonal arrays with considering interaction effects could also be investigated in particle swarm optimization. The resulting algorithm could be used to model those systems or processes which are highly dimensional and nonlinear.

Acknowledgments

The work described in this paper is supported substantially by a grant from the Research Grants Council of Hong Kong, China (Project No. PolyU 5184/07).

References

1. Aydin M.E. and Fogarty T.C. (2004). A distributed evolutionary simulated annealing for combinatorial optimisation problems, Journal of Heuristics, 10(3), 269-292.

2. Box G.E.P., Hunter W.G. and Hunter J.S. (1978). Statistics for Experimenters. John Wiley.

3. Bryne D.M. and Taguchi S. (1986). The Taguchi approach to parameter design, *ASQC Quality Congress Transaction*, Anaheim, 168.

4. Cerny V. (1985). Thermodynamical approach to the travelling salesman problem: an efficient simulation algorithm. Journal of Optimization Theory and Applications, 45, 41–51.

5. Chatterjee S., Carrera C. and Lynch L.A. (1995). Genetic algorithms and travelling salesman problems. European Journal of Operational Research, 93, 490-510.

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

7. Fogel D.B. (1994). An introduction to simulated evolutionary optimization. IEEE Transactions of Neural Networks, 5(1), 3-14.

8. Gong G., Liu Y. and Qian M. (2001). An adaptive simulated annealing algorithm. Stochastic Processes and their Applications, 94, 95-103.

9. Ho S.J., Ho S.Y., and Shu L.S. (2004). OSA: Orthogonal simulated annealing algorithm and its application to designing mixed H_2=H_∞ Optimal Controllers. IEEE Transactions on Systems. Man and Cybernetics – Part A: Systems and Humans, 34(5), 588-600.

10. Ho S.Y., Ho S.J., Lin Y.K. and Chu W.C.C. (2004). An orthogonal simulated annealing algorithm for large floorplanning problems. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 12(8), 874-876.

11. Ho S.Y., Shu L.S. and Chen J.H. (2004). Intelligent evolutionary algorithms for large parameter optimization problems. IEEE Transactions on Evolutionary Computation, 8(6), 522-541.

12. Ho S.J., Shu L.S. and Ho S.Y. (2006). Optimizing fuzzy neural networks for tuning PID controllers using an orthogonal simulated annealing algorithm OSA. IEEE Transactions on Fuzzy Systems, 14(3), 421-434.

13. Kim J.D. and Choi M.S. (1995). Stochastic approach to experimental analysis of cylindrical lapping process. International Journal of Machines Tools Manufacturing, 35(1), 51-59.

14. Kirkpatrick S., Gelatt JR., and Vecchi M.P. (1983). Optimization by simulated annealing. Science, 220, 671-680.

15. Kratica J., Tosic D., Filipovic V. and Ljubic I. (2001). Solving the simple plant location problem by genetic algorithm. RAIRO Operations Research, 35, 127-142.

16. Kwong C.K., Chen Y. and Bai H. Integrating perceptual product mapping with QFD for new product design. (working paper)

17. Lin C.K.Y., Haley K.B. and Sparks C. (1995). A comparative study of both standard and adaptive versions of threshold accepting and simulated annealing algorithms in three scheduling problems. European Journal of Operational Research, 83, 330-346.

18. Lin Y.H., Tyan Y.Y., Chang T.P. and Chang C.Y. (2004). An assessment of optimal mixture for concrete made with recycled concrete aggregates. Cement and Concrete Research, 34, 1373-1380.

19. Locatelli M. (2000). Simulated annealing algorithms for continuous global optimization: convergence conditions. Journal of Optimization Theory and Applications, 104(1), 121-133.

20. Mohan N.S., Ramachandra A. and Kulkarni S.M. (2005). Influence of process parameters on cutting force and torque during drilling of glass fiber polyester reinforced composites. Composite Structures, 71, 407-413.

21. Moilanen A. (2001). Simulated evolutionary optimization and local search: introduction and application to tree search. Cladistics, 17, 12-15.

22. Phadke M.S. (1987). Quality engineering using robust design. New York: Prentic Hall.

23. Reeves C.R. and Wright C.C. (1995). An experimental design perspective on genetic algorithms. Foundation of Genetic Algorithms, 3, 7-22.

24. Reeves C.R. and Wright C.C. (1995). Epistasis in Genetic Algorithms: An Experimental Design Perspective. Proceedings of the 6th International Conference on Genetic Algorithms (pp. 217-224).

25. Reeves C.R. (1999). Predictive measures for problem difficulty. Proceedings of the 1999 Congress on Evolutionary Computation, 1, (pp. 736-742).

26. Ruiz-Torres A.J., Enscore E.E. and Barton R.R. (1997). Simulated annealing heuristics for the average flow-time and the number of Tardy jobs bi-criteria identical parallel machine Problem. Computers Industry Engineering, 33(1-2), 257-260.

27. Shu L.S., Ho S.Y. and Ho S.J. (2004). A novel orthogonal simulated annealing algorithm for optimization of electromagnetic problems. IEEE Transactions on Magnetics, 40(4), 1791-1795.

28. Szu H. and Hartley R. (1987). Fast simulated annealing. Physical Letters, 122, 157-162.

29. Szu H. (1987). Nonconvex optimization by fast simulated annealing. Proceedings of the IEEE, 75(11), 1538-1540.

30. Tsallis C. and Stariolo D.A. (1996). Generalized simulated annealing. Physica A, 233, 395-406.

31. Unal R., Stanley D.O. and Joyner C.R. (1993). Propulsion system design optimization using the Taguchi Method. IEEE Transactions on Engineering Management, 40(3), 315-322.

32. Van Laarhoven P.J.M., Aarts E.H. and Lenstra J.K. (1992). Job shop scheduling by simulated annealing. Operations Research, 40(1), 113-125.

33. Van Laarhoven P.J.M. and Aarts E.H.L. (1987). Simulated Annealing: Theory and Applications. D. Reidel Publishing Co.

34. Vaessens R.J.M., Aarts E.H.L. and Lenstra J.K. (1992). A local search template. Proceedings of Parallel Problem-Solving from Nature (pp. 65-74).

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

36. Wong S.Y.W. (2001). Hybrid simulated annealing/genetic algorithm approach to short term hydro-thermal scheduling with multiple thermal plants. Electric Power Energy Systems, 23, 565-575.

37. Yao X., Lin Y. and Lin G. (1999). Evolutionary programming made faster. IEEE Transactions on Evolutionary Computation, 3(2), 82-102.

38. Yao X. (1991). Simulated annealing with extended neighbourhood. International Journal of Computer Mathematics. 40, 169-189.

39. Yao X. (1993). Comparison of different neighbourhood sizes in simulated annealing. Proceedings of 4th Australian Conference on Neural Networks (pp. 216-219).

40. Yin G.Z. and Jillie D.W. (1989). 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, (181-198).

Appendix 1

Algorithm Q=ONF(P_1)

Step 1) Generate P_2 and P_3 with $P_1 = \left(S_1, ..., S_m\right)$ based on (4).

Step 2) Divide P_1, P_2 and P_3 into N groups based on (5).

Step 3) Represent levels 2, 1 and 3 of the j-th parameter of $L_{2N+1}\left(3^N\right)$ by the j-th group of P_1, P_2 and P_3 respectively.

Step 4) Compute y_t based on the t-th combination of $L_{2N+1}\left(3^N\right)$ as the t-th experiment.

Step 5) Compute the main effect M_{jk} where j=1,...,N and k=1,2,3 based on (1).

Step 6) Determine the best level *Best*(j) on the j-th parameter based on (2) for smaller-the-better problems and (3) for larger-the-better problems.

Step 7) The candidate solution Q is produced by the combinations of best levels of parameters.

Appendix 2

[Insert Tables 2 to 8 here]

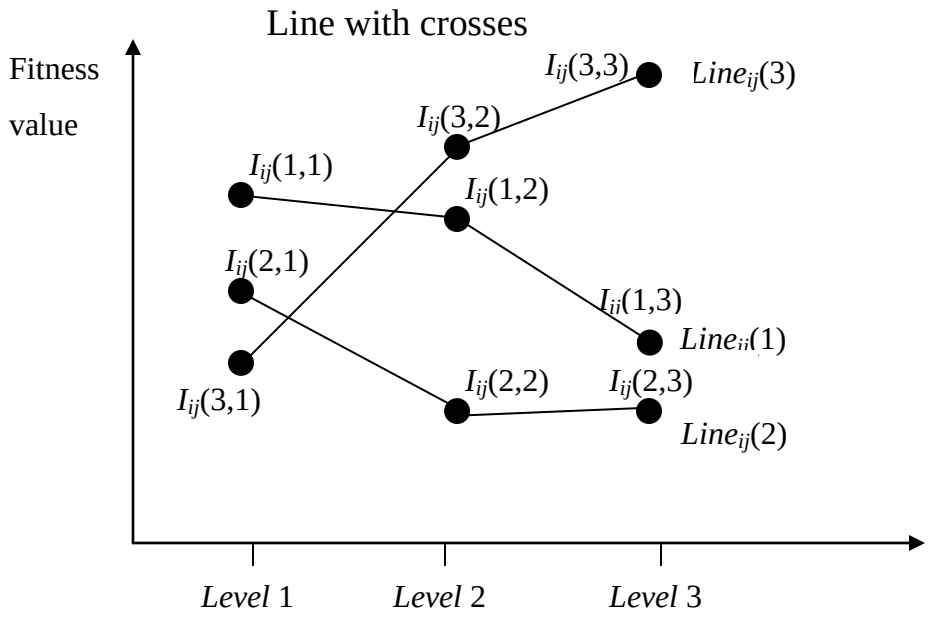

Figure 1 Strong interaction effect exists between parameters *i* and *j*

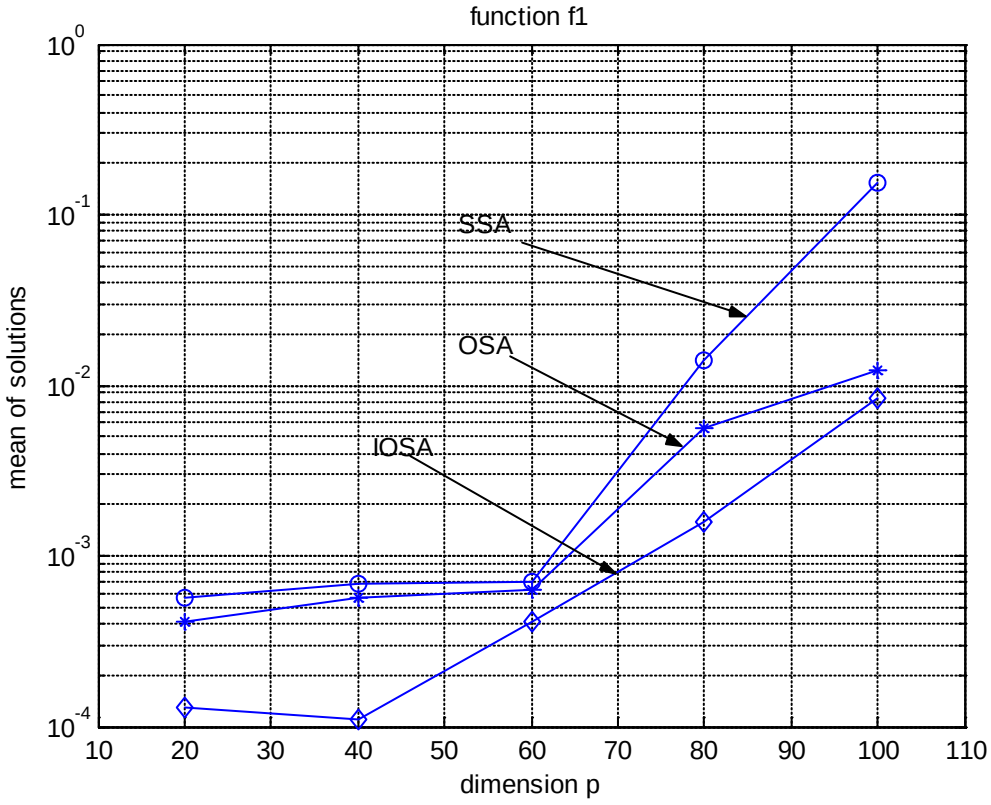

Figure 2 Results of function f_1

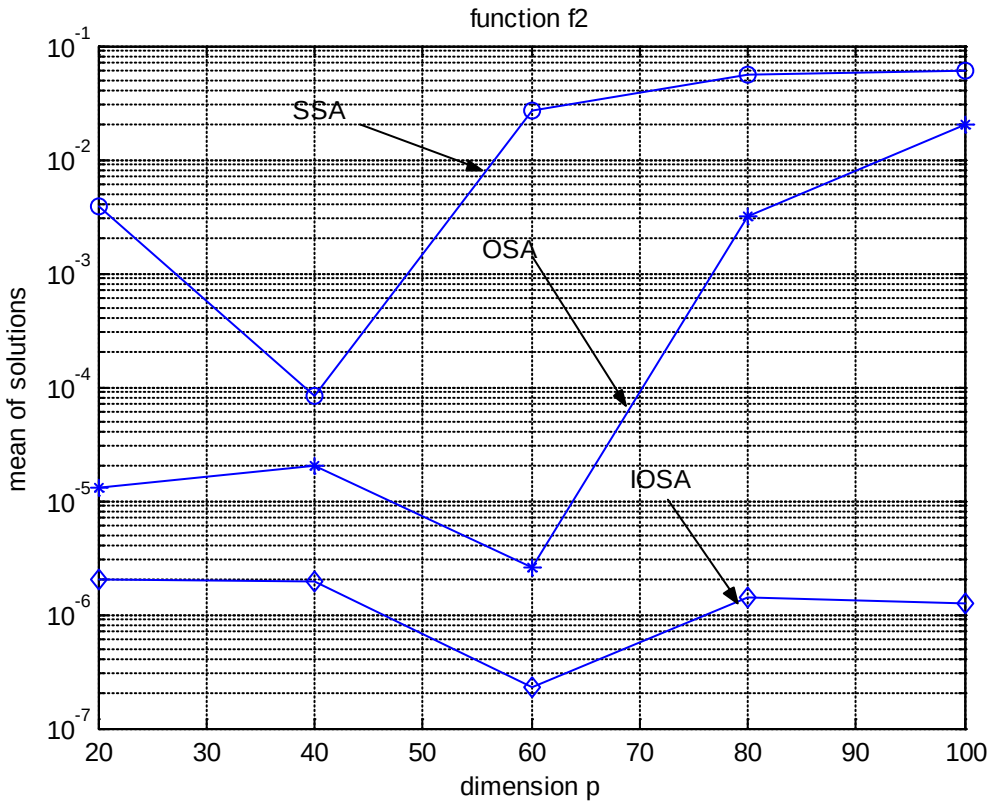

Figure 3 Results of function f_2

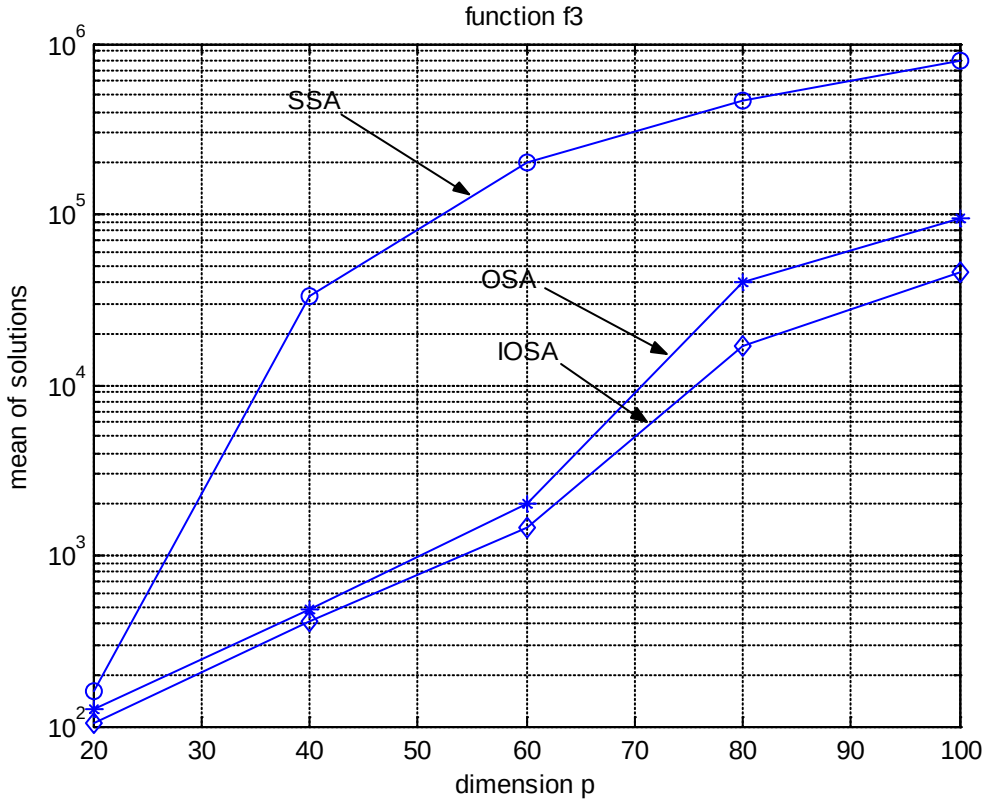

Figure 4 Results of function f_3

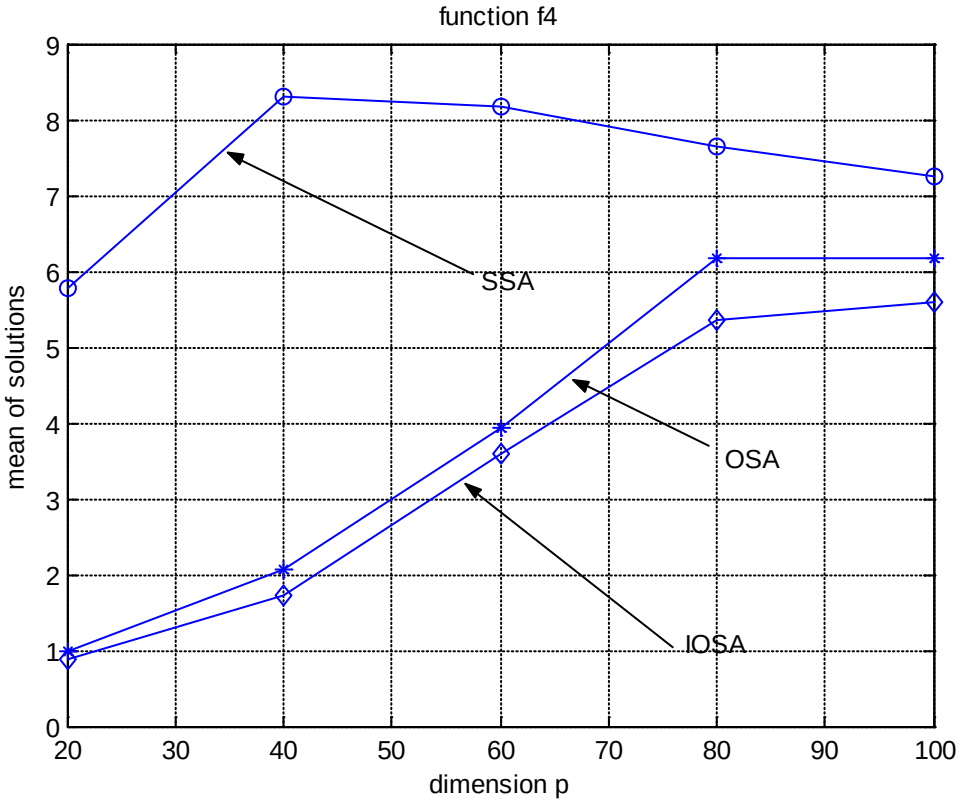

Figure 5 Results of function f_4

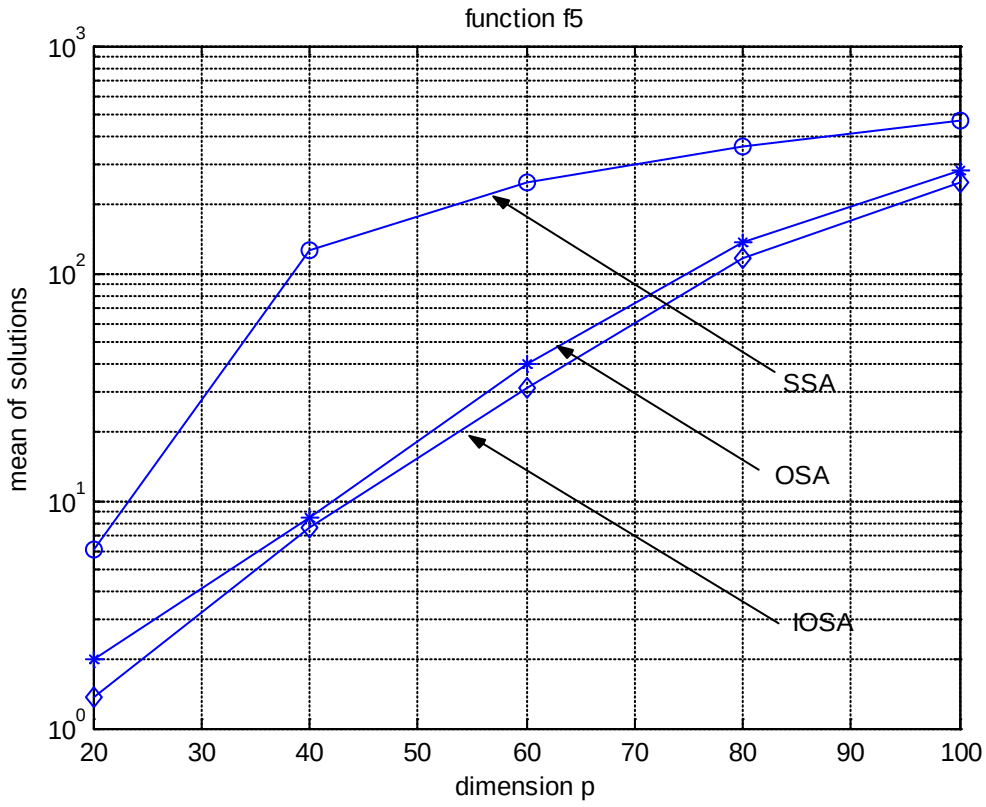

Figure 6 Results of function f_5

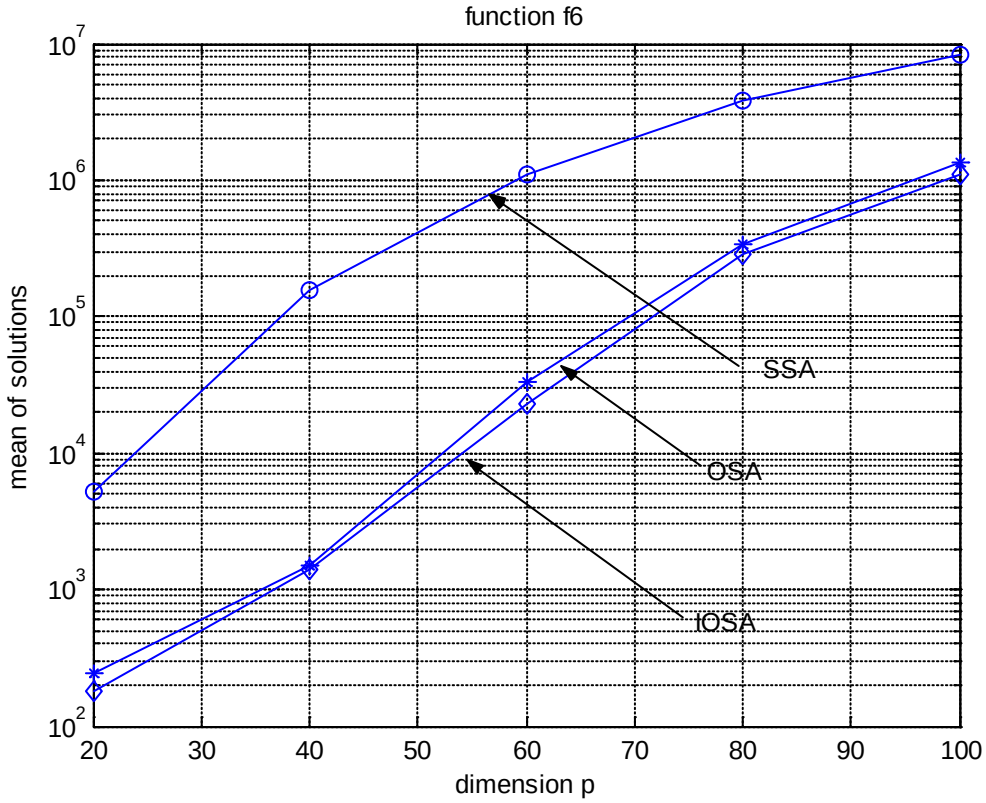

Figure 7 Results of function f_6

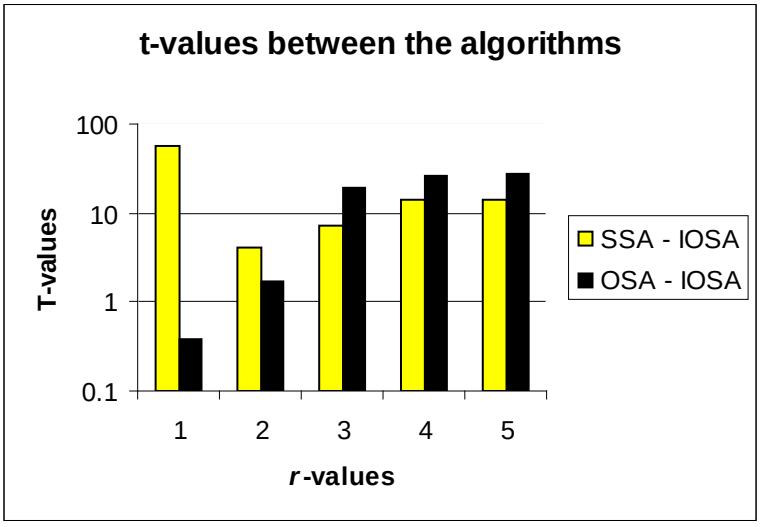

Figure 8 *t*-values between the algorithms

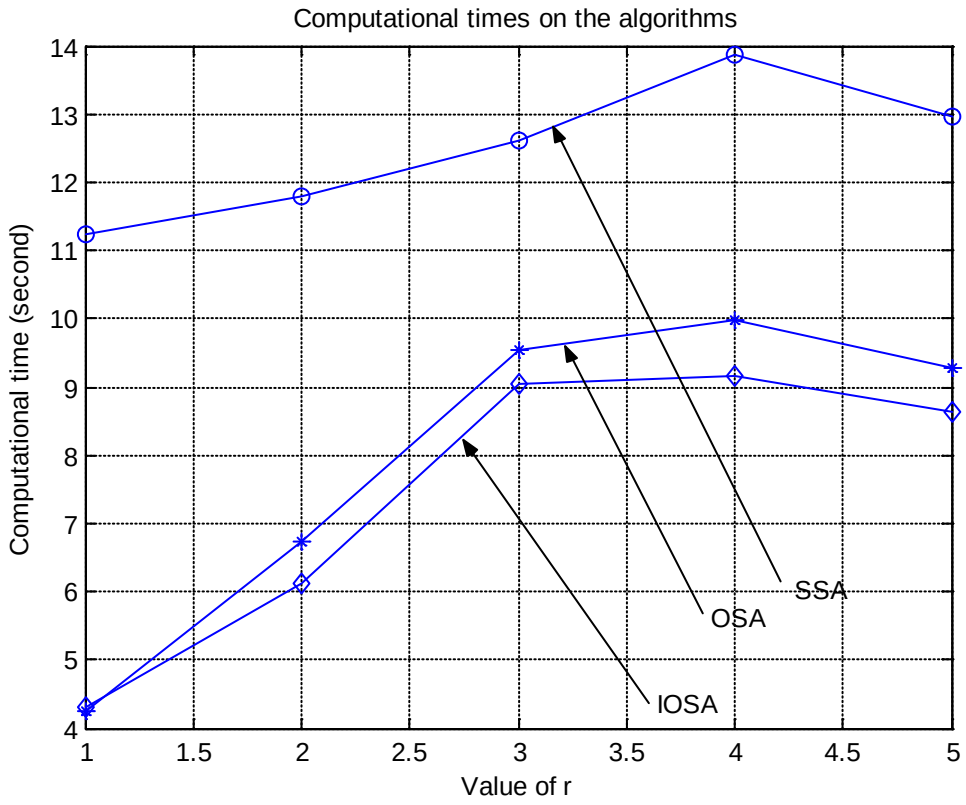

Figure 9 Computational time on solving the emulsified dynamite packing-machine design problem

Table 1 Selected test suites

Parametrical benchmark functions	Domain range (x_i)	Minimum
$f_1 = \min\left\{\sum_{i=1}^{P} 2N + \sum_{i=1}^{P-1}\left[\sin(x_i + x_{i+1}) + \sin\left(\frac{2x_i x_{i+1}}{3}\right)\right]\right\}$	$[3,13]^P$	0
$f_2 = \min\left(\frac{1}{4000}\sum_{i=1}^{P} x_i^{\,2} - \prod_{i=1}^{P}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1\right)$	$[-600,600]^P$	0
$f_3 = \min\sum_{i=1}^{P-1}\left[100\left(x_{i+1} - x_i^{\,2}\right)^2 + (x_i - 1)^2\right]$	$[-5.12,5.12]^P$	0
$f_4 = \min\left(20 + e - 20e^{-0.2\sqrt{\frac{\sum_{i=1}^{P} x_i^{\,2}}{N}}} - e^{\frac{\sum_{i=1}^{P}\cos(20\pi x_i)}{N}}\right)$	$[-30,30]^P$	0
$f_5 = \min\left(\sum_{i=1}^{P}\lvert x_i\rvert + \prod_{i=1}^{P}\lvert x_i\rvert\right)$	$[-10,10]^P$	0
$f_6 = \min\sum_{j=1}^{P}\left(\sum_{i=1}^{j} x_i\right)^2$	$[-100,100]^P$	0

Table 2 Result of benchmark tests based on f_1

N=	20	40	60	80	100
SSA mean	0.5701×10^{-3}	0.6776×10^{-3}	0.7053×10^{-3}	0.1411×10^{-1}	0.1522×10^{0}
SSA std	0.6554×10^{-3}	0.5836×10^{-3}	0.6530×10^{-3}	2.7248×10^{-3}	0.1494
OSA mean	0.4088×10^{-3}	0.5686×10^{-3}	0.6393×10^{-3}	0.5546×10^{-2}	0.0123×10^{0}
OSA std	0.1086×10^{-3}	0.4831×10^{-3}	0.5799×10^{-3}	1.7035×10^{-3}	0.9814×10^{-1}
IOSA mean	0.1284×10^{-3}	0.1105×10^{-3}	0.4091×10^{-3}	0.1562×10^{-2}	0.0085×10^{0}
IOSA std	0.0173×10^{-3}	0.0854×10^{-3}	0.4887×10^{-3}	0.4785×10^{-3}	0.4959×10^{-1}

std – standard deviation

Table 3 Result of benchmark tests based on f_2

N=	20	40	60	80	100
SSA mean	0.3879×10^{-2}	0.8310×10^{-4}	0.0269	0.0549119	0.0610416
SSA std	0.6680×10^{-2}	1.9315×10^{-4}	0.1473	0.2037	0.2206
OSA mean	0.0013×10^{-2}	0.1989×10^{-4}	0.2635×10^{-5}	0.003210	0.02036
OSA std	0.4111×10^{-3}	0.2368×10^{-5}	07335×10^{-5}	1.7452×10^{-4}	0.1010
IOSA mean	0.0002×10^{-2}	0.0192×10^{-4}	0.2286×10^{-6}	1.410343×10^{-6}	1.25755×10^{-6}
IOSA std	0.4765×10^{-4}	0.2503×10^{-5}	0.5935×10^{-6}	1.7535×10^{-6}	1.0850×10^{-6}

std – standard deviation

Table 4 Result of benchmark tests based on f_3

N=	20	40	60	80	100
SSA mean	1.61998×10^{2}	3.322986×10^{4}	2.01267×10^{5}	4.5998×10^{5}	7.9039×10^{5}
SSA std	1.02351×10^{2}	2.036671×10^{4}	7.195423×10^{4}	1.006506×10^{5}	1.37904×10^{4}
OSA mean	1.262328×10^{2}	4.853939×10^{2}	2.02662×10^{3}	4.04657×10^{4}	9.57764×10^{4}
OSA std	42.70728	1.99286×10^{2}	1.35091×10^{3}	2.61012×10^{4}	3.84730×10^{4}
IOSA mean	1.046856×10^{2}	4.09799×10^{2}	1.45172×10^{3}	1.6685×10^{4}	4.599238×10^{4}
IOSA std	30.61629	1.31064×10^{2}	1.095457×10^{3}	1.32495×10^{4}	1.923225×10^{4}

std – standard deviation

Table 5 Result of benchmark tests based on f_4

N=	20	40	60	80	100
SSA mean	5.78800	8.31013	8.18400	7.6318988	7.238934
SSA std	2.20272	0.69861	0.349668	0.24445	0.2487
OSA mean	0.9788	2.06892	3.93421	6.18335	6.170734
OSA std	0.1888	0.58577	0.75633	0.52325	0.3433
IOSA mean	0.8939	1.727303	3.60402	5.36611	5.60388
IOSA std	0.17321	0.343023	0.659474	0.57294	0.3425

std – standard deviation

Table 6 Result of benchmark tests based on f_5

N=	20	40	60	80	100
SSA mean	6.10265	1.26377×10^{2}	2.53108×10^{2}	3.62345×10^{2}	4.673949×10^{2}
SSA std	4.66434	25.6713	23.0232	30.8571	30.6804
OSA mean	1.98888	8.42111	39.66437	1.36615×10^{2}	2.8576×10^{2}
OSA std	0.48300	1.70283	8.56296	28.47988	31.3114
IOSA mean	1.37129	7.59070	31.42920	1.17040×10^{2}	2.5174×10^{2}
IOSA std	0.362736	1.424418	9.39414	34.64055	28.96643

std – standard deviation

Table 7 Result of benchmark tests based on f_6

N=	20	40	60	80	100
SSA mean	5.19732×10^{3}	1.56938×10^{5}	1.11410×10^{6}	3.83418×10^{6}	8.24746×10^{6}
SSA std	1.1171×10^{4}	6.9159×10^{4}	2.73481×10^{5}	7.26947×10^{6}	1.09917×10^{6}
OSA mean	2.39963×10^{2}	1.52578×10^{3}	3.27441×10^{4}	3.42513×10^{5}	1.34091×10^{6}
OSA std	1.24722×10^{2}	2.3918×10^{2}	2.71025×10^{4}	1.22372×10^{5}	5.93870×10^{5}
IOSA mean	1.82887×10^{2}	1.4093×10^{3}	2.321762×10^{4}	2.87125×10^{5}	1.11755×10^{6}
IOSA std	1.0915×10^{2}	1.8154×10^{2}	1.28995×10^{4}	1.14011×10^{5}	3.04895×10^{5}

std – standard deviation

Table 8 Results of validation tests based on the case of design optimization

r=	1	2	3	4	5
SSA mean	4.1223	3.5952	4.0196	4.0130	3.9232
SSA std	0.1223×10^{-3}	0.4756	0.1509×10^{-0}	0.8520×10^{-1}	0.1190×10^{-1}
OSA mean	4.2372	4.0089	4.1333	4.1246	4.0010
OSA std	0.9903×10^{-6}	0.4792	0.9413×10^{-1}	0.1658×10^{-1}	0.3740×10^{-1}
IOSA mean	4.2373	4.3098	4.2372	4.2368	4.2373
IOSA std	$0.\ 2679\times10^{-5}$	0.4793	0.6072×10^{-1}	0.1391×10^{-1}	0.1264×10^{-1}

std – standard deviation