A Study on Search Algorithms for Constructing Optimal Designs

Jaratsri Rungrattanaubol[1], Anamai Na-udom[1*]

Received: 18 February 2013 Accepted: 15 June 2013

Abstract

Computer simulated experiments (CSE) are often used in science and engineering applications. The nature of CSE is that they are time consuming and computationally expensive to run. Normally, the output response from computer simulated experiments is deterministic. Consequently the space filling designs, which focus on spreading design points over a design space, are necessary. Latin hypercube designs (LHD) are normally practiced in the context of CSE. The optimal LHD for a given dimension of problem is constructed by using a search algorithm under a pre-specified optimality criterion. Usually this searching process takes a long time to terminate, especially when the dimension of the problem is large. This paper proposes methods to enhance the performance of search algorithms which are widely used in the context of CSE. The comparative studies are employed based on a range of problems and optimality criteria. The results indicate that the proposed method can improve the capability of the search algorithms for constructing the optimal LHD.

Keywords: Computer simulated experiments, Latin hypercube designs, simulated annealing algorithms, enhanced stochastic evolutionary algorithm, optimality criteria

Introduction

Recently computer simulated experiments (CSE) have replaced classical experiments to investigate a physical complex phenomena, especially when classical (physical) experiments are not feasible. For example, the use of reservoir simulator to predict ultimate recovery of oil, the use of finite element codes to predict behavior of metal structure under stress, and so on[1]. The nature of computer simulated experiments is deterministic[2,3] hence identical settings of input variables always produce an identical set of output response. Therefore, space filling designs that aim to spread the design points over a region of interest are necessary. The most popular class of space filling design in the context of computer simulated experiments is Latin hypercube design (LHD). LHD design was originally proposed by Mckay and co-workers[4] in 1979. The ultimate goal of selecting the settings of input variables is to attain the coverage of all design regions of interest.

As mentioned before the space filling designs are preferred in the context of computer simulated experiments. Space filling designs or the optimal LHD can be constructed through combinatorial methods (non-search algorithm)[5,6] or searching for a design through search algorithms[7,8]. The former method generates design with good design properties but it is restricted in terms of a design size. For example methods proposed by Butler[5] are limited to a design size of a prime number. The latter method is based largely on improving design by exchanging between the pairs of design points. Exchange algorithms can be time consuming to implement, however, the generated design are flexible and straightforward. The CSEs are usually complex and consist of many input variables to investigate[9]. In this case a large number of runs are required to estimate the parameters corresponding to the factors of interest in the model. For example, if the problem of interest consists of d input variable and n number of runs, the total number of LHD is $(n!)^d$. Obviously

[1] *Department of Computer Science and Information Technology,* [2] *Department of Mathematics, Faculty of Science Naresuan University Phitsanulok, Thailand*

** Corresponding author: anamain@nu.ac.th*

this number explodes exponentially as the values of n and d increase; hence the full space of LHD cannot be explored. In this case we need the search algorithms to lead us to a good design with respect to an optimality criterion. The key idea of all existing search algorithms is to use some kinds of exchange procedures to move towards the better designs.

The search based approach for selecting a design is implemented by combining search algorithms and the optimality criterion[8]. For example, Morris and Mitchell[7] adopted a version of Simulated Annealing algorithms (SA) to search for optimal LHDs with respect to ϕ_p criterion. Li and Wu[10] proposed a columnwise-pairwise algorithm (CP) with respect to the *D* efficiency criterion. It was reported that CP is very simple and easy to implement. The only parameter required to set as a priori is the tolerance level (*tl*). Further, CP is able to generate a good supersaturated design and it can be used along with various optimality criteria[7]. In order to avoid the problem of convergence and the search being stuck at a local optimum value, usually multiple search with different starting points are performed. The best result, among different trials, is selected as optimal design. It should be noted that for large dimensional problems, CP algorithm can be time consuming to implement. Ye and his co-workers[6] adapted CP algorithm to search for symmetric LHD under various optimality criteria such as entropy and ϕ_p criteria. Park[8] proposed a row-wise element exchange algorithm along with IMSE and entropy criteria. Leary et al[11] adapted CP and SA algorithms to construct the optimal designs within the orthogonal-array based Latin hypercube class by using the ϕ_p criteria. Jin et al.[12] developed an enhanced stochastic evolutionary algorithm (ESE) to search for the best design considering various optimality criteria such as a maximin distance criterion, ϕ_p criterion and entropy criterion. ESE has received wide attention from researchers due to its performance in constructing the optimal LHD. Liefvendahl and Stocki[13] applied a version of Genetic algorithm (GA) to search for the optimal LHD considering ϕ_p and a maximin distance criterion. A similar work can be found in[14] as the authors applied GA for constructing maximin designs. Grosso et al.[15] used the iterated local search algorithm and SA in constructing the optimal LHD under maximin distance and ϕ_p criterion. Vianna et al.[16] proposed the algorithm for fast optimal LHD by using the idea of seed design under maximin distance and ϕ_p criterion. Due to the popularity of SA and ESE along with ϕ_p criteria, this paper presents the efficient method to improve the capability of SA and ESE under ϕ_p criterion. In the following sections we present details of these search algorithms, followed by the details of the optimality criteria. The enhancement methods on SA and ESE are also presented in section III. The results of the enhancement methods will be presented in the result section and conclusion will be given in section V respectively.

Experimental Design and Optimality Criterion

This section presents the details of LHD and the steps of search algorithms including the enhancement methods to improve their performance in constructing the optimal LHD.

Latin hypercube design (LHD)

LHD can be constructed based on the idea of stratified sampling[4] to ensure that all subregions in the divided input variable space will be sampled with equally probability. A Latin hypercube sampling has

$$X_{ij} = \frac{\pi_{ij} - U_{ij}}{n} \qquad (1)$$

where π_{ij} are the elements of an $n \times d$ matrix comprising of columns $\pi_j (j = 1, 2, \ldots, d)$. Each column $\pi_j (j = 1, 2, \ldots, d)$ is independent random permutation of number 1 through n and U_{ij} are $n \times d$ values of independent *U[0,1]* random variables independent of the π_{ij}.The example of LHD is shown in Figure 1.

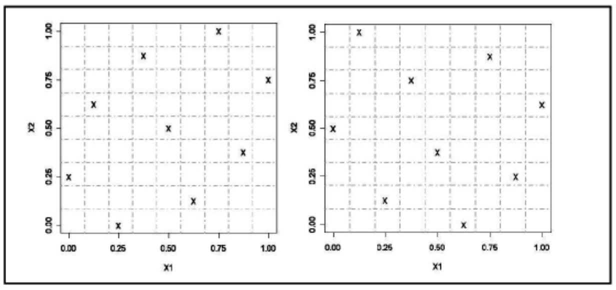
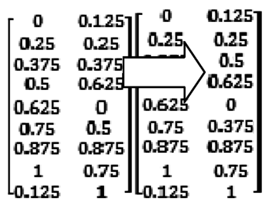


Figure 1 Example of 9×2 LHD

Element exchange operation

The element exchange operation to construct a new LHD design is developed by using the concept of column-wise operation proposed by Li and Wu[10]. The process is randomly interchange two distinct elements in a randomly selected column as shown in Figure 2. After an element exchange has been performed, the LHD properties still remains.



$$\begin{bmatrix} 0 & 0.125 \\ 0.25 & 0.25 \\ 0.375 & 0.375 \\ 0.5 & 0.625 \\ 0.625 & 0 \\ 0.75 & 0.5 \\ 0.875 & 0.875 \\ 1 & 0.75 \\ 0.125 & 1 \end{bmatrix} \Rightarrow \begin{bmatrix} 0 & 0.125 \\ 0.25 & 0.25 \\ & 0.5 \\ & 0.625 \\ 0.625 & 0 \\ 0.75 & 0.375 \\ 0.875 & 0.875 \\ 1 & 0.75 \\ 0.125 & 1 \end{bmatrix}$$

Figure 2 Element exchange in the 2nd column of a 9×2 LHD

The ϕ_p optimality criterion

Morris and Mitchell[12] proposed a modification class of maximin distance criterion to search for the optimal design. For a given design X, the Euclidean intersite distance between any two design points can be calculated from

$$d(x_i, x_j) = \left[\sum_{k=1}^{d} (x_{ik} - x_{jk})^2 \right]^{1/t} \tag{2}$$

By using (2), all intersite distances for every pairs of design points are calculated and can be expressed in the symmetric matrix form as follows.

$$D = \begin{bmatrix} d_{11} & d_{12} & \cdots & d_{1j} \\ d_{21} & d_{22} & \cdots & d_{2j} \\ \vdots & \vdots & \ddots & \vdots \\ d_{i1} & d_{i2} & \cdots & d_{ij} \end{bmatrix}$$

Let a Euclidean distance list $(d_1, d_2, \ldots, d_m)$ be the distinct elements list from the smallest to largest. Also define index list $(J_1, J_2, \ldots, J_m)$, which J_j is the number of pairs of sites in the design separated by distance d_j. Thus X is a maximin design if among available designs, it maximizes d_j while J_j is minimized. The scalar criterion can be expressed as $\phi_p = [\sum_{j=1}^{m} J_j d_j^{-p}]$ (3)

, where p is a positive integer, J_j and d_j specified from X. The design that minimizes ϕ_p is a maximin LHD in the class. In this study, the adaptive form of ϕ_p[4] which is simpler than (3) to implement is considered

$$\phi_p = \left[\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{1}{d_{ij}^p} \right]^{\frac{1}{p}} \tag{4}$$

After ϕ_p value has been calculated, a design that minimizes ϕ_p is considered as an optimal design in the class.

Search algorithms

This section presents the details of search algorithms used in this study including the enhancement methods to improve the performance of the search algorithms.

Modification of simulated annealing algorithm (MSA)

Morris and Mitchell[7] developed a simulated annealing algorithm to search for an optimal LHD using ϕ_p optimality criterion. The design that minimizes ϕ_p value is considered as the best design in the class. The steps of SA are presented as follows.

```
Step 1: Set initial values
    $I_{\max}$ (maximum number of perturbation to seek improvement)
    $t_0$ (initial cooling temperature)
    $C_t$ (factor by which $t_0$ is reduced when no improvement in $\phi_p$)
    $n$ (Number of runs)
    $d$ (Number of input variables)
Step 2: Generate a random $LHD$, $X$ of given order $n \times d$.
    Let $X_{best} = X, t = t_0$
Step 3: Set $I = 1, Label = 0$
Step 4: Let $X_{try} = X$
    Randomly select a column say j, of matrix $X_{try}$ and exchange two randomly selected elements of column j, say $X_{aj} \leftrightarrow X_{bj}$
Step 5: Set $X = X_{try}, Label = 1$ ,If $\phi_p(X) - \phi_p(X_{try}) \ge tl$
    or with probability $e^{-\left[\phi_p(X_{try}) - \phi_p(X)\right]/t}$
Step 6: If $\phi_p(X_{try}) < \phi_p(X_{best})$, set $I = 1$ and $X_{best} = X_{try}$,
        else $I = I + 1$.
Step 7: If $I < I_{\max}$ go to Step 4.
Step 8: If $Lable = 1$, set $t = t \times C_t$, go to Step 3.
Step 9: Stop and report $X_{best}$.
```

SA requires parameter settings, t_0, $I_{\max}$, FAC_t and p. In this study, we use the heuristic methods to find the best set of parameters for use in SA. The choice of initial parameters for SA can be found in[7]. It was also reported in the paper that SA performed very well in terms of moving away from the local optimum value of ϕ_p criterion.

The emphasis of this paper is on the modification of SA by applying the calculation of ϕ_p criterion by using the method that avoids re-calculating of ϕ_p value. As mentioned before, SA uses the exchange procedure between two pairs of points within the randomly selected column. Hence, after an exchange between rows i_1 and i_2 within column k $(x_{i_1k} \leftrightarrow x_{i_2k})$, only elements in rows i_1 and i_2, and columns i_1 and i_2 are changed in the distance matrix D[4].

For any $1 \le j \le n$ and $j \ne i_1, i_2$ let:

$$s(i_1,i_2,k,j) = \left|x_{i_2k} - x_{jk}\right|^t - \left|x_{i_1k} - x_{jk}\right|^t \tag{5}$$

then

$$d'_{i_1j} = d'_{ji_1} = \left[d^t_{i_1j} + s(i_1,i_2,k,j)\right]^{1/t} \tag{6}$$

and

$$d'_{i_2j} = d'_{ji_2} = \left[d^t_{i_2j} + s(i_1,i_2,k,j)\right]^{1/t} \tag{7}$$

Thus new ϕ_p is computed by

$$\phi'_p = \left[\begin{array}{l}\phi_p^p + \sum\limits_{1\le j\le n, j\ne i_1,i_2} \left[(d'_{i1j})^{-p} - (d_{i1j})^{-p}\right] + \\ \sum\limits_{1\le j\le n, j\ne i_1,i_2} \left[(d'_{i2j})^{-p} - (d_{i2j})^{-p}\right]\end{array}\right]^{1/p} \tag{8}$$

As shown in (5) to (8), only some rows and columns are updated to calculate ϕ_p criterion in MSA. Hence the complexities or BigO of MSA is much smaller than SA as presented in Table 1.

Table 1 The complexities to calculate ϕ_p criterion in SA and MSA

BigO(SA)	$O(dn^2) + O(n^2 \log_2(p))$
BigO(MSA)	$O(n) + O(n\log_2(p))$

ESE and modification of ESE algorithm (MESE)

As presented in the previous section, the complexity of MSA is less than the original SA. Hence MSA is recommended for use in constructing the optimal design for CSE if time constraint is of interest. Jin et al.[12] proposed a new algorithm called enhanced stochastic evolutionary (ESE) algorithm and did a comparison between ESE and the existing algorithms such as CP and SA. The results showed that ESE is superior over the other algorithms in terms of computational time burden and the number of exchanges required for generating the optimal LHD design. According to the goodness of MSA and ESE, we combine them together to improve the search process. The In next section we present the steps of ESE including the methods to improve the performance of ESE.

ESE was developed from the stochastic evolutionary (SE) algorithm proposed by Sabb and Rao[17]. It contains 2 nested loop called inner and outer loops. The inner loop performs a local search process by constructing a new design and decides whether to accept new design or not. In the inner loop, both of acceptance ratio and improvement ratio are recorded. The outer loop works as a controller of the inner loop as it performs a global search by adjusting the threshold (Th) based on acceptance ratio and improvement ratio from the inner loop. The steps of ESE are presented as follows.

Step 1: Set initial parameters and design X_0 in the outer loop, $X = X_0$, $i = 0$, $n_{act} = 0$ and $n_{imp} = 0$
Step 2: Construct a set of new design X_{try}
Step 3: Select the best design X_{try} from this set
Step 4: Decide to accept the best design X_{try} and replace the current best design X from as shown in Figure 3.
Step 5: If X_{try} is better than the global best design X_{best}, replace it with X_{try} and increase n_{imp} by 1 ($n_{imp} = n_{imp} + 1$).
Step 6: Terminate the inner loop if $i > M$, else go to step 2.

The flowchart of the inner loop for ESE is visualized in Figure 3

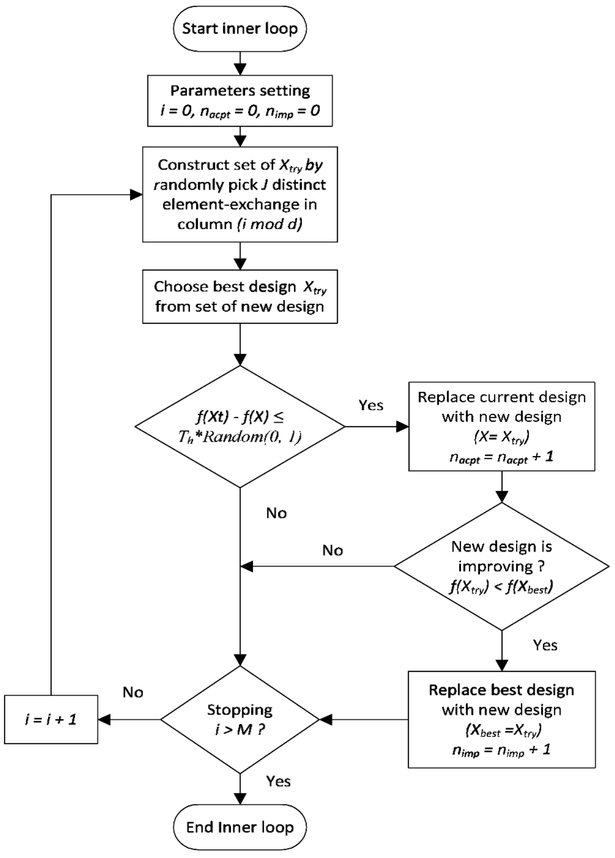

Start inner loop
Parameters setting
i = 0, n_acpt = 0, n_imp = 0
Construct set of X_try by randomly pick J distinct element-exchange in column (i mod d)
Choose best design X_try from set of new design
f(Xt) - f(X) ≤ T_h*Random(0, 1)
Yes
No
Replace current design with new design (X= X_try) n_acpt = n_acpt + 1
New design is improving ? f(X_try) < f(X_best)
No
Yes
Replace best design with new design (X_best =X_try) n_imp = n_imp + 1
Stopping i > M ?
No
i = i + 1
Yes
End Inner loop

Figure 3 Flowchart of ESE inner loop[1]

In this study the parameters J is set to be $\binom{n}{2}/5$ but no larger than 50, and the parameter M is in a range of $2\times\binom{n}{2}\times d/J \le M \le 100$. The outer loop of ESE is presented in Figure 4.

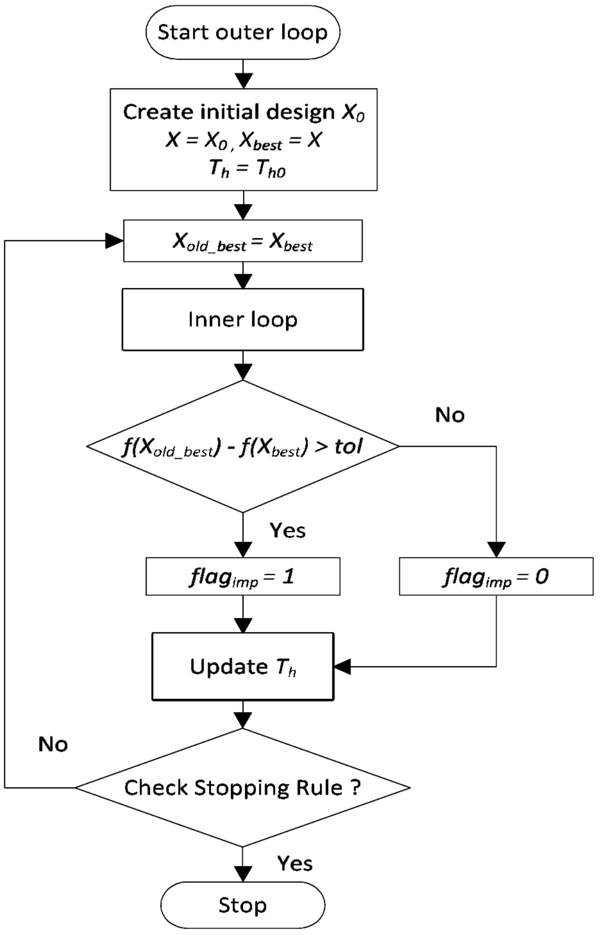


Figure 4 Flowchart of ESE outer loop[12]

The details of outer loop are given below.

Step 1: Randomly generate an initial design X_0 and set $X = X_0$, $X_{best} = X$, initialize $Th_0 = 0.005 \times \phi_p(X_0)$ and $Th = Th_0$.

Step 2: Set $X_{old_{best}} = X_{best}$.

Step 3: Go to the inner loop process.

Step 4: Select a method to update Th, by setting $flag_{imp}$.

Step 5: Update Th (discussed later for more details).

Step 6: Terminate the search by using a stopping rule, else go to step 2.

The tolerance level (*tl*) is set to 0.0001 as it was observed from the empirical study that a smaller value does not improve the search process. The process of updating the value of Th in step 5 is divided into 2 processes called improving process and exploration process, respectively. The search process works as the improving process when $flag_{imp} = 1$, if the best design X_{best} is improved in the inner loop. If not, the search process will be in the exploration process ($flag_{imp} = 0$). In improving process ($flag_{imp} = 1$), Th is adjusted in order to find the local best LHD based on an acceptance ratio (n_{act} / p)and improvement ratio (n_{imp} / M). If $n_{act} / M > \beta_1$ and $n_{imp} / M < n_{act} / M$, then Th is decreased by $T_h = \alpha_1 \times Th$. If If $n_{act} / M > \beta_1$ and $n_{imp} / M = n_{act} / M$, then Th is unchanged. Otherwise, T_h is increased by $Th= Th / \alpha_1$, where $0 < \alpha_1 < 1$ and $0 < \beta_1 < 1$, we use $\alpha_1 = 0.8$ and $\beta_1 = 0.1$ as suggested by Jin et al. [4]. Further the results obtained from our empirical studies also indicate that β_1 should be set to a small value. In the exploration process ($flag_{imp} = 0$), T_h will be adjusted to drive the algorithm to move far away from a local optimal design based on the range of accept ratio. If $n_{act} / M < \beta_2$, then T_h is increased until $n_{act} / M > \beta_3$ by equation $Th= Th / \alpha_3$. If $n_{act} / M > \beta_3$, then T_h is decreased till $n_{act} / M < \beta_2$ by equation $Th= Th \times \alpha_2$, where $0 < \beta_2 < \beta_3 < 1$, and $0 < \beta_2 < \beta_3 < 1$, we set $\alpha_2 = 0.9$, $\alpha_3 = 0.7$. While β_2 should be small, we set $\beta_2 = 0.1$ and β_3 should be large enough so we set $\beta_3 = 0.8$, as recommended in[4].

Modification of ESE (MESE)

In this section we present the enhancement method on ESE. The modified version is called MESE. We combine the advantage of SA (i.e. local search process) and the advantage of ESE (i.e. global search process) together to improve the search process. MESE contains 2 nested loops as displayed in Figure 5. The outer loop is similar to the ESE except that there is only one change in a stopping rule as in step 6. The maximum number of cycles used is replaced by the following condition. If a local best design after the inner loop X_{best} is not improved from the global best design ($X_{globalbest}$) for δ consecutive times, then the search process will be terminated. In this study we set $\delta = 10$.

The major enhancement was made in the inner loop. There are many changes have been made in step 2, step 5 and step 6. In step 2, the process for constructing a new design X_{try} is changed to element-exchange in column (*i* mod *d*) for all *J* iterations while the original ESE used the random strategy to pick *J* distinct element-exchange in column (*i* mod *d*). By doing this, the computational complexity decreases from $O(n^2)$ to $O(n)$. As can be seen in ESE process, a random element exchange for all *J* iteration is required in all i^{th} iterations, so all distinct $i-1$ loops must be checked. Hence the complexity is $O(n(n-1)) = O(n^2)$. In MESE, we adapt the process of element-exchange from SA shown in Figure 6.

So in any *J* iteration, element exchange of a current design *X* in column *i* mod *d* is independent. Thus there is no need to perform all *J* iterations. It is obvious that the computation complexity decreases to $O(n)$. In step 5, if a new design X_{try} is improved (better than the best achieved design, X_{best}), let *j* = 0 otherwise increase *j* by 1 (*j*= *j* + 1). Finally, in step 6 of the inner loop, a stopping rule is modified to if *i* > *M* or *j* > $C_{\max}$. In this study, we set $C_{\max} = 10$. All simulation studies presented in this paper were performed using R program[18].

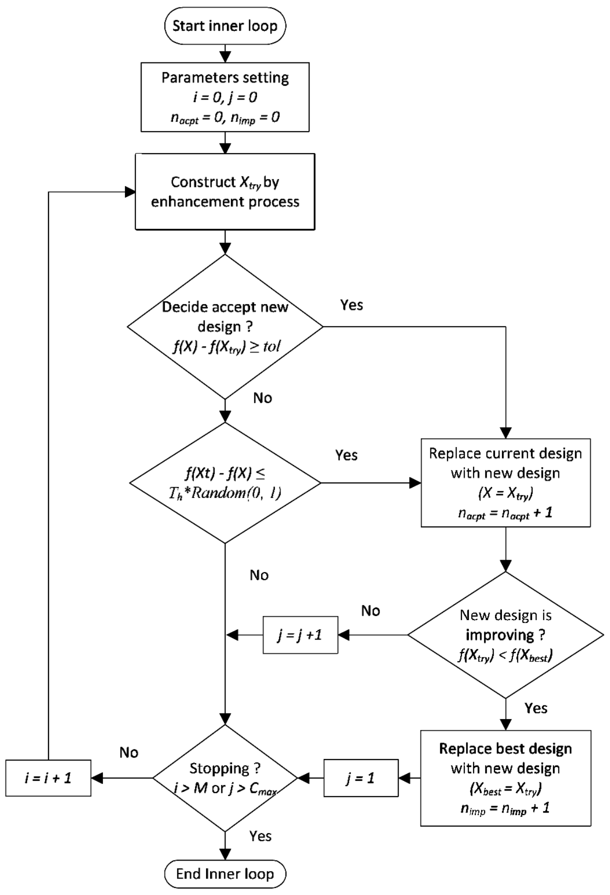


Figure 5 Flowchart of MESE inner loop

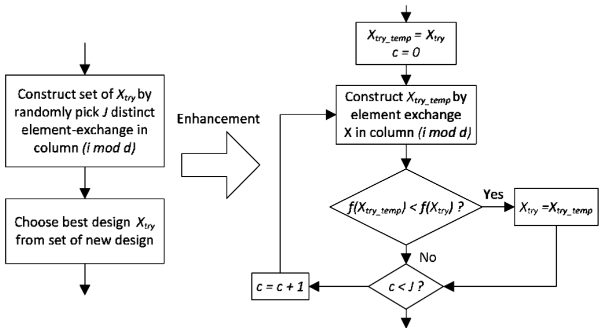

Construct set of X_try by randomly pick J distinct element-exchange in column (i mod d)
Choose best design X_try from set of new design
Enhancement
X_try_temp = X_try
c = 0
Construct X_try_temp by element exchange X in column (i mod d)
f(X_try_temp) < f(X_try) ?
Yes
No
X_try = X_try_temp
c < J ?
c = c + 1

Figure 6 A new design construction using SA (Step 2)

Results

The values of ϕ_p criteria at the termination step of MSA, ESE and MESE from each dimension of problems are presented in Table 2. Each case study was repeated for 10 times to consider the effect of different starting points. The descriptive statistics on the ϕ_p values obtained from each search technique are displayed in columns 3-6. The results in columns 3-6 indicate that MSA, ESE and EMSE perform similarly for small dimension of problem in terms of minimization of ϕ_p criterion. Further, the standard deviation values appeared in column 6 displays a slightly larger amount of variation over 10 replications in ESE and EMSE than that of MSA. This indicates the consistency in the search process for MSA when different starting points are considered. When medium dimensions are considered, $\phi_p$values from ESE and MESE are slightly lower than MSA. In addition, small amount of standard deviation is observed. For large dimensions of problem, both of ESE and MESE perform similar results in terms of minimization of ϕ_p values. Hence if the good property of design is concerned, either ESE or MESE can be used for constructing the optimal LHD.

Table 2 Descriptive statistics of ϕ_p values obtained from MSA, ESE and MES

		(p = 5, t = 2)			
LHDs	Algorithm	Min	Max	Mean	SD
9 × 2	MSA	4.273	4.273	4.273	0
	ESE	4.273	4.344	4.287	0.029
	MESE	4.273	4.344	4.280	0.022
51 × 5	MSA	5.417	5.43	5.422	0.004
	ESE	5.415	5.431	5.422	0.006
	MESE	5.418	5.43	5.423	0.004
201 × 10	MSA	6.179	6.181	6.180	0.000
	ESE	6.170	6.174	6.172	0.001
	MESE	6.173	6.185	6.184	0.000
451 × 15	MSA	6.776	6.779	6.777	0.001
	ESE	6.760	6.762	6.761	0
	EESE	6.760	6.762	6.761	0
801 × 20	MSA	7.272	7.273	7.272	0
	ESE	7.253	7.254	7.254	0
	EESE	7.254	7.254	7.254	0

The results of the performance (efficiency) for MSA, ESE and MESE algorithms are presented in Table 3. This table presents the time elapsed and number of exchange required for each algorithm to reach the same level of ϕ_p values. For each dimension of problem, the search algorithms are repeated for 10 times. Hence all values are presented as the average values. For small dimension case, it can be clearly seen that ESE and MESE converges much faster than MSA. The number of exchange required in the search process is also less than the MSA, while MESE requires less number of exchanges comparing to ESE. For medium and large dimensions of problem, MESE converges much faster than MSA while it performs slightly better than ESE. Further, the number of exchange obtained from MESE is the smallest value. This indicates that if time constraint is taken into account, MESE could be the better choice to use in the construction of the optimal LHD designs.

The results in columns 5-7 display time ratio for each search algorithm. It can be concluded from these ratio that MESE converges much more quickly than MSA. The maximum improvement over MSA can be observed when the dimension of problem is small. In the case of larger dimension, the improvement ratio turns to a small value. It could be concluded that the performance of these three algorithms are close to each other especially ESE and MESE algorithm.

Table 3 Performance of MSA, ESE and EMSE

LHDs	**Algorithm**	**Performance (Average)**		**Time ratio**		
		Time (sec.)	**#Exchange**	**MSA/ ESE**	**MSA/ MESE**	**ESE/MESE**
2 × 9	MSA	19.993	47140	5.561	8.707	1.565
	ESE	3.595	5760			
	MESE	2.296	5415			
51 × 5	MSA	751.54	284931	1.364	2.400	1.759
	ESE	550.762	150000			
	MESE	313.067	118950			
201 × 10	MSA	2795.741	209912	2.071	2.376	1.147
	ESE	1349.788	17580			
	MESE	1176.529	94070			
451 × 15	MSA	8686.660	234517	2.380	2.463	1.034
	ESE	3648.646	185220			
	MESE	3526.691	124550			
801 × 20	MSA	20854.01	260539	2.692	2.718	1.009
	ESE	7744.424	220480			
	MESE	7672.487	185750			

Conclusions

This paper presents a method to enhance the SA and ESE algorithms in the construction of the optimal LHD. The major enhancement method appears in the calculation of ϕ_p criterion and the tolerance level setting in SA. For ESE, the enhancement is applied by using the combination of SA and ESE especially in the inner loop as shown in Figure 5 and 6. As presented in the result section, MESE perform better than ESE and MSA in terms of the design property achievement and the efficiency. Hence MESE would be recommended for the construction of optimal LHD for CSE. In order to extend the conclusion, other classes of design can be developed and collaborated with MESE to search for the best design in the class. Further, other types of search algorithm like Particle swarm optimization (PSO) or Ant colony can be further developed in constructing an optimal LHD or other classes of space filling design. The validation of the approximation model accuracy developed from the obtained optimal design could also be further investigated.

References

1. J. Koehler, and A. B. Owen, “Computer experiments. Handbook of Statistics”, vol. 13, Elsevier Science, New York, 1996, 261-308.
2. J. Sacks, W. J. Welch, T.J. Mitchell, and H. P. Wynn, “Design and analysis of computer experiments,” Statistical Science, vol. 4(4), 1989, 409-435.
3. W. J. Welch, R. J. Buck, J. Sacks, H. P Wynn, T. J. Mitchell, and M. D. Morris, “Screening, predicting, and computer experiments,” Technometrics, vol. 34, 1992, 15-25.
4. M.D. Mackay, R. J. Beckman, and W. J. Conover, “A comparison of three methods for selecting values of input variables in the analysis of output from a computer code,” Technometrics, vol. 21, 1979, 239-246.
5. N. A. Butler, “Optimal and orthogonal latin hypercube designs for computer experiments”, Biometrika, 88(3), 2001, 847-857.
6. K. Q. Ye, W. Li, and A. Sudjianto, “Algorithmic construction of optimal symmetric latin hypercube designs,” Journal of Statistical Planning and Inference, vol. 90, 2000, 145-159.
7. M.D. Morris, and T. J. Mitchell, “Exploratory design for computational experiments,” Journal of Statistical Planning and Inference, vol. 43, 1995, 381-402.
8. J. S. Park, “Optimal latin hypercube designs for computer experiments,” Journal of Statistical Planning and Inference, vol. 39, 1994, 95-111.
9. R. A. Bates, R. J. Buck, E. Riccomagno, and H. P. Wynn, “Experimental design and observation for large systems”. Journal of the royal statistical society, Series B, 58, 1996, 77-94.
10. W. Li, and C.F.J. Wu. “Columnwise-pairwise algorithms with applications to the construction of supersaturated designs” Technometrics, vol. 39, 1997, 171-179.
11. S. Leary, A. Bhaskar and A. Keane, “Optimal orthogonal -array-based latin hypercubes”. Journal of Applied Statistics, vol. 30(5), 2003, 585-598.
12. R. Jin, W. Chen, and A. Sudjianto, “An efficient algorithm for constructing optimal design of computer experiments”. Journal of Statistical Planning and Inference, vol. 134, 2005, 268-287.
13. M. Liefvandahl and R. Stocki, “Study on algorithms for optimization of latin hypercubes,” Journal of Statistical Planning and Inference, vol. 136, 2006, 3231-3247.
14. Z. Li and N. Shigeru, “Maximin distance-lattice hypercube design for computer experiment based on genetic algorithm,” *IEEE explore*, vol. 2, 2001, 814-819.
15. A. Grosso, A. Jamali, and M. Locatelli, “Finding maximin latin hypercube designs by iterated local search heuristics”, European Journal of Operation Research, vol. 197(2), 2009, 541-547.
16. F. A. C. Viana, G. Venter, V. Balanov, “An algorithm for fast optimal latin hypercube design of experiments,” International Journal for Numerical Methods in Engineering, vol. 82(2), 2010, 135-156.
17. Y. G. Saab, and Y. B. Rao, “Combinatorial optimization by stochastic evolution,” IEEE Transaction on Computer-aided Design, vol.10, 1991, 525–535, 1991.
18. R Development Core Team (2011). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.R-project.org/.