Stat Comput (2013) 23:663–676
DOI 10.1007/s11222-012-9363-3

Optimizing Latin hypercube designs by particle swarm

Ray-Bing Chen · Dai-Ni Hsieh · Ying Hung · Weichung Wang

Received: 30 November 2011 / Accepted: 13 October 2012 / Published online: 23 October 2012

Abstract Latin hypercube designs (LHDs) are widely used in many applications. As the number of design points or factors becomes large, the total number of LHDs grows exponentially. The large number of feasible designs makes the search for optimal LHDs a difficult discrete optimization problem. To tackle this problem, we propose a new population-based algorithm named LaPSO that is adapted from the standard particle swarm optimization (PSO) and customized for LHD. Moreover, we accelerate LaPSO via a graphic processing unit (GPU). According to extensive comparisons, the proposed LaPSO is more stable than existing approaches and is capable of improving known results.

Keywords Latin hypercube design · Particle swarm optimization · Graphic processing unit (GPU)

1 Introduction

The study of computer experiments has received significant attention in the past decade (Kennedy and O'Hagan 2000; Santner et al. 2003). Because many of these experiments are time consuming, it is important to design the experiments carefully with a small number of experimental runs. To explore an experimental region efficiently, the experimental design in a computer experiment is usually required to satisfy two properties. First, the design should be space-filling to acquire the maximum amount of information on the experimental domain. Second, the design should be non-collapsing. That is, the design does not have two points that only differ in one factor (or variable). This property is necessary because if this factor is not significant, the responses of the two points will be nearly identical and will offer little information. Hence, a collapsing design will result in a waste of computation, especially for a deterministic model. Based on these two properties, a space-filling Latin hypercube design (LHD) is an appropriate and popular choice.

An n-run and k-factor LHD is constructed as follows. Each factor is divided into n levels, and the design points are arranged such that each level has exactly one point. Thus, a LHD has the one-dimensional projection property (Santner et al. 2003), which means that projecting an n-point design onto any factor leads to exactly n different levels for that factor. Formally, an n-run and k-factor LHD can be represented as an n by k matrix. Each row of the matrix corresponds to a design point, and each column is a permutation of $\{1, 2, \ldots, n\}$. The total number of n-run and k-factor LHDs is $(n!)^{k-1}$, which grows rapidly as n or k increases. Although a LHD is guaranteed to be non-collapsing, a randomly generated LHD may not always be space-filling. Figure 1(a) shows an example of a non-space-filling LHD that concentrates its design points in the diagonal part of the experimental region while leaving a large area unexplored.

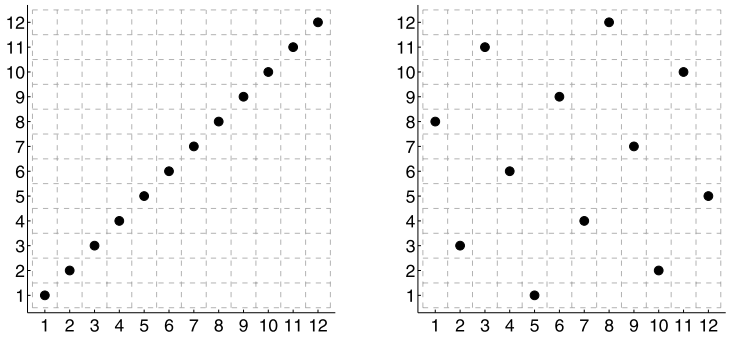
To achieve the space-filling property, different criteria are used, such as the max-min criterion (Grosso et al. 2009; Liefvendahl and Stocki 2006; van Dam et al. 2007, 2009), the ϕ_p criterion (Grosso et al. 2009; Morris and Mitchell

R.-B. Chen
Department of Statistics, National Cheng Kung University, Tainan 701, Taiwan

D.-N. Hsieh
Institute of Statistical Science, Academia Sinica, Taipei 115, Taiwan

Y. Hung
Department of Statistics and Biostatistics, Rutgers University, Piscataway, NJ 08854, USA

W. Wang (✉)
Department of Mathematics, National Taiwan University, Taipei 106, Taiwan
e-mail: wwang@ntu.edu.tw



(a) A non-space-filling LHD (b) Max-min LHD

Fig. 1 Two LHD examples where $n = 12$ and $k = 2$

1995; Ye et al. 2000; Jin et al. 2005; Viana et al. 2010), the Audze-Eglais criterion (Liefvendahl and Stocki 2006; Bates et al. 2004), the entropy criterion (Ye et al. 2000), and the centered L_2-discrepancy (Fang et al. 2002). Although different criteria are adopted, the common challenge for these discrete optimization problems is that the huge number of feasible designs renders the search for the optimal LHDs as non-trivial, especially for large n or k. To tackle this issue and find the optimal LHDs efficiently, many optimization methods have been proposed. Morris and Mitchell (1995) presented a simulated annealing search algorithm and noted that some optimal LHDs have a symmetric structure. Following the observation of Morris and Mitchell (1995), Ye et al. (2000) considered restricting the search space to a subset called symmetric Latin hypercube designs in order to reduce the search time. They used the columnwise-pairwise (CP) exchange algorithm. Jin et al. (2005) proposed the enhanced stochastic evolutionary algorithm (ESE), which contains inner and outer loops. The inner loop is similar to the CP algorithm, while the outer loop is used to determine the acceptance threshold based on the performance of the inner loop. Grosso et al. (2009) used the Iterated Local Search heuristics which is composed of a local search and perturbation procedures. As for the population based methods, Bates et al. (2004) and Liefvendahl and Stocki (2006) proposed different versions of genetic algorithms. Particularly for the 2-factor LHDs, van Dam et al. (2007) constructed the optimal max-min LHD with respect to the max-min l^∞- and l^1-distance criteria. In the case of l^2-distance, they designed a branch-and-bound algorithm and found the optimal LHD for n up to 70.

In addition to the aforementioned methods, particle swarm optimization (PSO) (Kennedy and Eberhart 1995) is another alternative for optimization problems inspired by the social behavior of bird flocking or fish schooling. PSO is an efficient, heuristically stochastic, iterative procedure and PSO usually converges to a global optimum quickly. For additional convergence analysis of PSO, see van den Bergh (2006), van den Bergh and Engelbrecht (2002). Many variants of PSO have been developed and used to solve different types of optimization problems in various applications (Bratton and Kennedy 2007; Engelbrecht 2006). In statistics, some fundamental inference problems can also be formulated as global optimization problems and then solved by PSO. One example is the maximum likelihood estimation (MLE) of unknown parameters. Toala et al. (2011) studied the kriging parameter estimation in computer experiments, which requires the optimization of a multi-modal likelihood function, and they showed that a modified PSO approach is superior to other approaches in the search for MLE in kriging. Furthermore, PSO is capable of solving high-dimensional optimization problems with multiple optima, another attractive feature in the search for optimal LHDs with large n or k. Observing these properties of PSO, we are motivated to study the application of PSO to the search for optimal LHDs.

In this paper, we propose LaPSO, a variant of PSO, to solve the corresponding LHD optimization problem using the ϕ_p criterion. In addition, we use a graphic processing unit (GPU) to accelerate the proposed LaPSO, which allows us to tackle larger problems. For the cases we examine, the results of LaPSO are more stable than the results from existing methods. Moreover, LaPSO is able to verify the max-min and extended max-min LHDs in the literature and even improve some of them.

The remainder of this paper is organized as follows. Section 2 introduces the optimal criterion we consider and discusses the corresponding computational challenges. Section 3 reviews the standard PSO and then proposes LaPSO. Numerical results are reported and discussed in Sect. 4. We conclude the paper in Sect. 5.

2 The target LHD problem and its computational challenges

In this paper, we focus on the ϕ_p criterion introduced by Morris and Mitchell (1995). The ϕ_p criterion is deduced from the max-min criterion, which maximizes the minimum inter-site distance of a LHD. In other words, the max-min

criterion can be represented as

$$\max_{D}\left\{\min_{\forall j>i} d(\mathbf{x}_i,\mathbf{x}_j)\right\}, \tag{1}$$

where D is any n-run and k-factor LHD and $d(\mathbf{x}_i,\mathbf{x}_j)$ is the distance between design points $\mathbf{x}_i$ and $\mathbf{x}_j$. To further break ties among optimal max-min LHDs, Morris and Mitchell (1995) proposed the extended max-min criterion. Denote the sorted distinct inter-site distances from minimum to maximum by $d_1, d_2, \ldots, d_m$, and let J_i be the number of inter-site distances equal to d_i. The max-min criterion only maximizes d_1. The extended max-min criterion extends the max-min criterion as follows:

(1a) maximizes d_1, and among designs for which this is true,
(1b) minimizes J_1, and among designs for which this is true,
(2a) maximizes d_2, and among designs for which this is true,
(2b) minimizes J_2, and among designs for which this is true,
$\vdots$
(ma) maximizes d_m, and among designs for which this is true,
(mb) minimizes J_m.

Furthermore, to rank the competing designs using this criterion with a scalar value, Morris and Mitchell (1995) proposed the function

$$\phi_p = \left[\sum_{k=1}^{m} J_k d_k^{-p}\right]^{\frac{1}{p}}, \tag{2}$$

where p is a positive integer. They showed that the ϕ_p criterion ranks LHDs the same way as the extended max-min criterion does when p is large. Compared with the extended max-min criterion, the ϕ_p criterion is computationally simpler. The ϕ_p score can be easily calculated without ordering the inter-site distances. In addition, the ϕ_p criterion is widely used in the literature on the optimal LHD search (Grosso et al. 2009; Jin et al. 2005; Morris and Mitchell 1995; Viana et al. 2010; Ye et al. 2000). Therefore, it can provide a reasonable comparison with existing approaches, such as the genetic algorithm and the enhanced stochastic evolutionary algorithm. Properties of ϕ_p, such as the upper bound and lower bound of this criterion, are well studied (Joseph and Hung 2008). Optimizing LHDs under the ϕ_p criterion, we aim to solve the following optimization problem

$$\min_{D} \phi_p(D). \tag{3}$$

The optimization problem (3) is discrete because LHDs are formed by discrete design points.

The computational challenges to solve the optimization problem (3) are threefold. First, the total number of LHDs, $(n!)^{k-1}$, grows rapidly as n or k becomes large. Second, optimal LHDs are very few compared to the total number of LHDs. For example, Fig. 2 shows the distribution of ϕ_p values for two particular cases where we can evaluate all LHDs individually. The figure indicates that the number of optimal LHDs is very small and even the portion of near-optimal LHDs is small compared to the total number of LHDs. This observation additionally suggests that we cannot expect the random search to find satisfactory LHD. Third, as shown below, two "nearby" LHDs do not necessary have similar ϕ_p values.

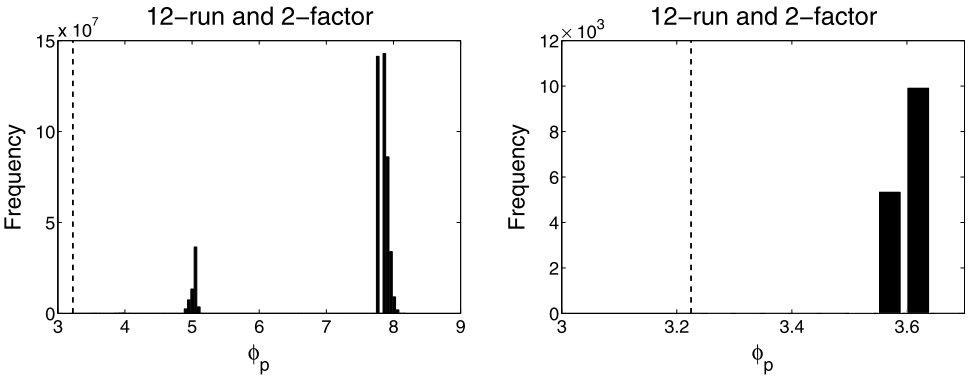
To measure the distance between two LHDs, or how close one LHD is to another LHD, we use the Hamming distance due to the nature of the problem. The Hamming distance is defined as the number of positions with different values between two LHDs. For example, consider the following two LHDs,

$$\begin{bmatrix} \underline{\mathbf{5}} & \underline{\mathbf{3}} & 4 \\ 2 & 4 & 3 \\ 3 & \underline{\mathbf{2}} & 1 \\ 1 & \underline{\mathbf{5}} & 2 \\ \underline{\mathbf{4}} & 1 & 5 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} \underline{\mathbf{4}} & \underline{\mathbf{5}} & 4 \\ 2 & 4 & 3 \\ 3 & \underline{\mathbf{3}} & 1 \\ 1 & \underline{\mathbf{2}} & 2 \\ \underline{\mathbf{5}} & 1 & 5 \end{bmatrix}.$$

The Hamming distance of these LHDs is five.

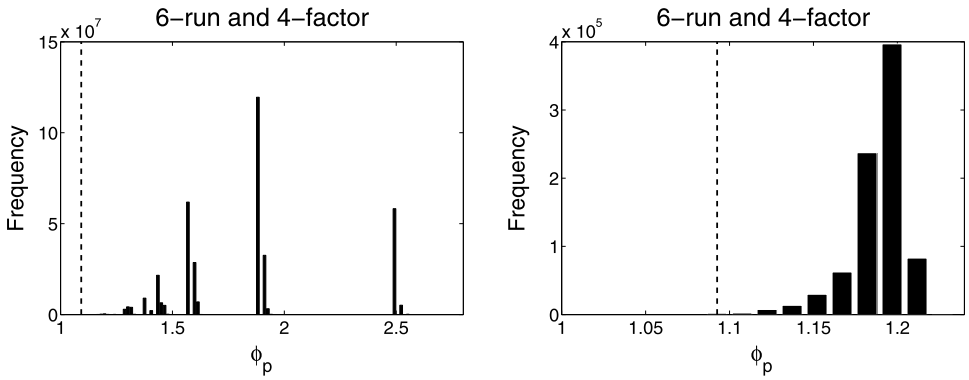
In the continuous domain, the values of a continuous function are near each other if two points are sufficiently close. This continuity is not the case for ϕ_p with respect to the Hamming distance. Consider the LHDs shown in Fig. 3. LHDs D_2 and D_3 in Figs. 3(b) and 3(c) are LHDs after a swap of the LHD D_1 in Fig. 3(a), so the Hamming distances between D_1 and D_2 and between D_1 and D_3 are both two. However, the corresponding ϕ_p value is similar for D_1 and D_2 but changes drastically from D_1 to D_3. We can see the reason in Fig. 3. A swap in the LHD may have a small impact on the overall structure, but a swap may cause some points to become clustered (or, from the opposite direction, may cause the points to spread).

We take a closer look at the relation between the ϕ_p value and the Hamming distance. Let us consider all LHDs with a Hamming distance two to the LHD D_1 in Fig. 3(a). We plot the probability distribution of the absolute difference of ϕ_p values between those LHDs and D_1. Similarly, we plot the distributions for other Hamming distances to D_1. The results are in Fig. 4. All figures show that absolute differences of ϕ_p above three account for a major portion in each plot. However, the ratio of LHDs with an absolute difference of ϕ_p below one increases as the Hamming distance decreases. This trend is especially clear for the distributions in which the Hamming distance is between two and four. In short, two LHDs separated by a smaller Hamming distance are not guaranteed to have closer ϕ_p values, but they have a higher chance of obtaining closer ϕ_p values.



(a) Histogram of ϕ_p with respect to all LHDs.

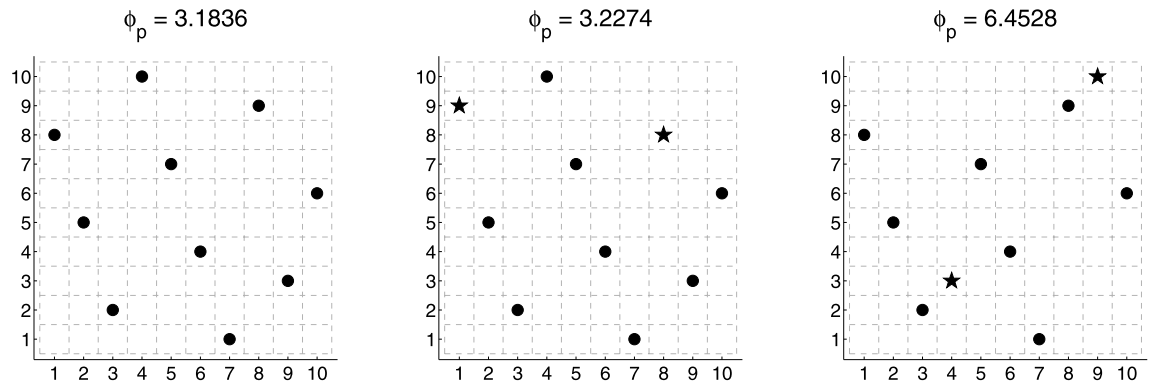
(b) Magnification of the left lower part of (a).



(c) Histogram of ϕ_p with respect to all LHDs.

(d) Magnification of the left lower part of (c).

Fig. 2 Distribution of ϕ_p values of LHDs ($p = 50$). The x- and y-axis represent the ϕ_p value and frequency, respectively. In addition, $(n, k) = (12, 2)$ for (**a**) and (**b**) and $(n, k) = (6, 4)$ for (**c**) and (**d**). The dashed line indicates the minimum. The number of LHDs with the minimum value is 2 for $(n, k) = (12, 2)$ and 192 for $(n, k) = (6, 4)$



(a) D_1 (b) D_2 (c) D_3

Fig. 3 The ϕ_p values ($p = 50$) of the LHD pairs with a Hamming distance of two. (**a**) A 10-run 2-factor LHD with $\phi_p = 3.1836$. (**b**) A small change of ϕ_p after a swap of the LHD D_1. ϕ_p becomes 3.2274. (**c**) A jump of ϕ_p after a swap of the LHD D_1. ϕ_p becomes 6.4528

3 Particle swarm optimization for Latin hypercube designs

After a brief review of the standard PSO in Sect. 3.1, we propose using LaPSO to tackle the computational challenges of solving the discrete optimization problem (3) in Sect. 3.2.

3.1 PSO

PSO is a population-based, stochastic, heuristic methodology and is capable of solving high-dimensional optimization problems with multiple optima. Since PSO was proposed (Kennedy and Eberhart 1995), many variants of PSO

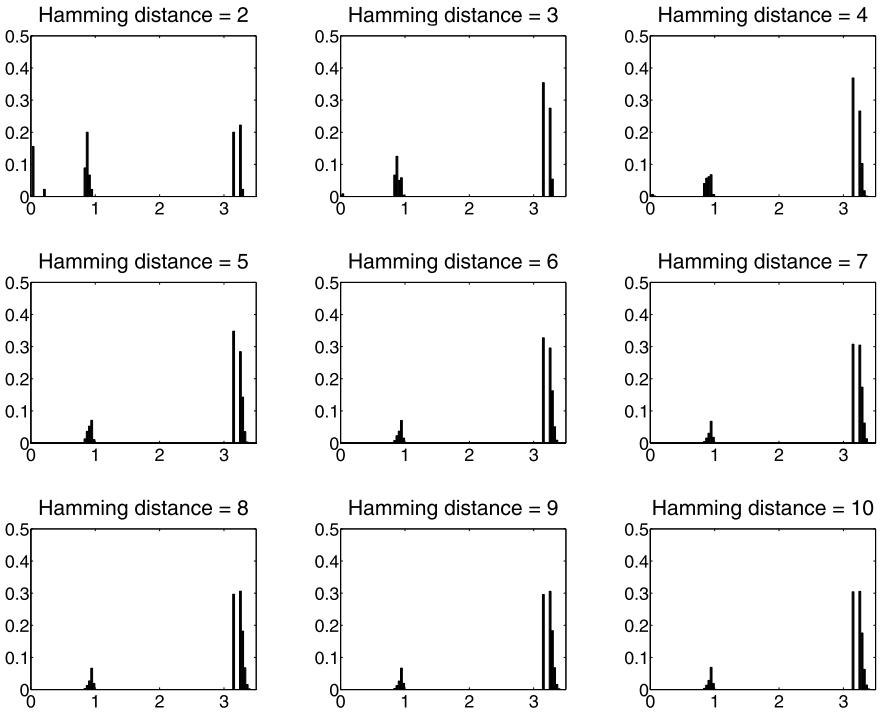


Fig. 4 The distributions of the absolute difference of ϕ_p ($p = 50$) between all LHDs and the LHD in Fig. 3(a) with respect to different Hamming distances. The x-axis is the absolute difference of ϕ_p, and the y-axis is the number of LHDs divided by the total number of LHDs in the corresponding plot, which can be treated as a probability. With a smaller Hamming distance, two LHDs have a higher probability of attaining similar ϕ_p values

Algorithm 1 PSO framework

```
Initialize positions of particles
Evaluate objective function value of each particle
Initialize personal best positions
Initialize the global best position
while not converge do
    Update velocity of each particle
    Update position of each particle
    Evaluate objective function value of each particle
    Update personal best positions
    Update the global best position
end while
```

have been developed and used to solve different types of optimization problems in various applications (Bratton and Kennedy 2007; Engelbrecht 2006).

We present the PSO framework to solve the following *continuous* minimization problem:

$$\min_{\mathbf{x}\in\mathcal{S}} f(\mathbf{x}), \quad \mathcal{S} = \left\{\mathbf{x}\in\mathbb{R}^N \mid l_j \le x_j \le u_j, j = 1,\ldots,N\right\} \tag{4}$$

where $f : \mathbb{R}^N \to \mathbb{R}$ is the objective function. The main structure of PSO is outlined in Algorithm 1. PSO maintains a swarm of particles that move in the search space. The position of each particle represents the search point at which the objective function f is evaluated. The initial positions of particles are chosen randomly over the search space. Then, the velocity of each particle is determined based on its searching history (cognitive component) and information from other particles (social component). Next, each particle moves according to its velocity. After moving to the new position, each particle evaluates its function value at this new position. Each particle iteratively updates its velocity and position. Throughout the process, all particles cooperate to find the minimum value by social interaction.

Specifically, the velocity and position of a certain particle are updated by the following formulas:

$$\mathbf{v}_i^{(t+1)} = \omega\mathbf{v}_i^{(t)} + c_1\boldsymbol{\beta}_1^{(t)}\left(\mathbf{b}_i^{(t)} - \mathbf{x}_i^{(t)}\right) + c_2\boldsymbol{\beta}_2^{(t)}\left(\mathbf{b}_g^{(t)} - \mathbf{x}_i^{(t)}\right), \tag{5}$$

$$\mathbf{x}_i^{(t+1)} = \mathbf{x}_i^{(t)} + \mathbf{v}_i^{(t+1)}, \tag{6}$$

where i is the index for the particle and the vector product is a component-wise multiplication. The variables $\mathbf{v}_i$ and $\mathbf{x}_i$ are the velocity and position of the ith particle. The parameter ω is the inertia weight that is used to adjust the extent to which the previous velocity affects the current velocity. We use $\mathbf{b}_i$ to denote the personal best position. That is, $f(\mathbf{b}_i)$ has the minimum value of all the positions that the ith particle has ever visited till the current iteration. We use $\mathbf{b}_g$ to denote the global best position, which means that $f(\mathbf{b}_g)$ has the minimum value among all the positions that all of the particles have visited as yet. Two random vectors $\boldsymbol{\beta}_1$ and $\boldsymbol{\beta}_2$ are used to increase diversity while exploring the search space. The elements of both $\boldsymbol{\beta}_1$ and $\boldsymbol{\beta}_2$ are independently chosen from a uniform distribution between zero and one. The constants c_1 and c_2 represent the cognitive and social

Table 1 The sparsity of LHD points in terms of the number of LHDs $(n!)^{k-1}$ and the number of grid points $n^{n(k-1)}$

n	k	No. of LHD points	No. of grid points	Ratio
8	3	1.63E+09	2.81E+14	5.78E−06
8	4	6.55E+13	4.72E+21	1.39E−08
8	5	2.64E+18	7.92E+28	3.34E−11
10	3	1.32E+13	1.00E+20	1.32E−07
10	4	4.78E+19	1.00E+30	4.78E−11
10	5	1.73E+26	1.00E+40	1.73E−14

learning factor respectively. The magnitude of c_1 and c_2 determines how each particle weights its own personal best and global best experience while moving in search of the minimum.

3.2 LaPSO

Unlike the problem (4) whose search domain is continuous, we search for the optimal LHD among the finite and discrete LHD points, so we cannot apply PSO directly to the LHD optimization problem. To determine how to move particles between the iterations, there are at least two approaches to consider: (i) Move the particles using the ordinary continuous PSO and then round particles to the nearest LHD. (ii) Follow only the structure of the PSO and redefine the operations such that the move is from one LHD to another LHD. We next discuss these two approaches.

The first approach is intuitive, but how to find the nearest LHD to a given point is problematic. Even if we can find the nearest LHD, the sparsity of the LHDs will cause this idea to fail. Comparing the number of LHDs, i.e. $(n!)^{k-1}$, and the number of grid points, i.e. $n^{n(k-1)}$, in the search space of the same dimension, we find that $\lim_{n\to\infty}(n!)^{k-1}/n^{n(k-1)} = 0$ for a given k, which means LHDs become sparser as the dimension grows. Table 1 shows that the ratios of the numbers of LHDs to the numbers of grid points are small, even for small n and k. In other words, the nearest LHD to a given point would be farther than expected. For this reason, we take the second approach.

We now describe LaPSO which is outlined in Algorithm 2. The framework of LaPSO is similar to Algorithm 1 except that the steps for velocity and position update are modified. We show the details on how we move from one LHD to another LHD in LaPSO according to the "Move" and "RandSwap" functions.

The move function The term $(\mathbf{b}_i - \mathbf{x}_i)$ in the PSO velocity update formula (5) attracts a particle toward the personal best position. Following this idea, we want a method that can move the current LHD to another LHD that is closer to the personal best LHD in the sense of the Hamming distance. Because each column of a LHD is a permutation,

Algorithm 2 LaPSO: PSO for LHD

```
Generate random LHDs
Compute function value of each LHD
Initialize personal best LHDs
Initialize the global best LHD
for each iteration do
    Update position of each LHD via the nested for-loop
    for each particle LHD do
        for each column j of LHD do
            Move(LHD(j), pBest(j), SameNumP)
            Move(LHD(j), gBest(j), SameNumG)
            if rand(0, 1)< ProbR then
                RandSwap(LHD(j))
            end if
        end for
    end for
    Compute function value of each LHD
    Update personal best LHDs
    Update the global best LHD
end for
```

we must maintain this structure during movement. Focusing on one column, we proceed as follows. For a column of the current LHD, we randomly choose a position in the column. To decrease the Hamming distance, we make the value of this position the same as the value of the personal best in the corresponding position by swapping the value in the random position with the target value in the column of the current LHD. Figure 5 illustrates this process. After this step, the Hamming distance decreases by one or zero between the updated LHD and the personal best. We can apply this process several times to distinct positions in a column of the current LHD and then increase the similarity between the updated LHD and the personal best. This process is denoted by Move($LHD(j)$, $pBest(j)$, $SameNumP$), which means that we apply the above procedure to distinct randomly chosen $SameNumP$ positions in the jth column of LHD, so the updated jth column of LHD will have at least $SameNumP$ positions possessing the same values as the jth column of $pBest$. Similarly, we have the process Move($LHD(j)$, $gBest(j)$, $SameNumG$) for the global best.

The RandSwap function The function RandSwap($LHD(j)$) randomly chooses two different positions in the jth column of LHD and then swaps the values. The RandSwap function is executed with probability $ProbR$ in Algorithm 2.

We make two remarks to conclude this section. First, the proposed LaPSO algorithm is not restricted to the ϕ_p criterion. It can be applied to other criteria, such as the max-min criterion and the extended max-min criterion. Second, because the move from one design to another is completely random and all the input factors are treated equally, LaPSO is completely independent of the order in which the inputs

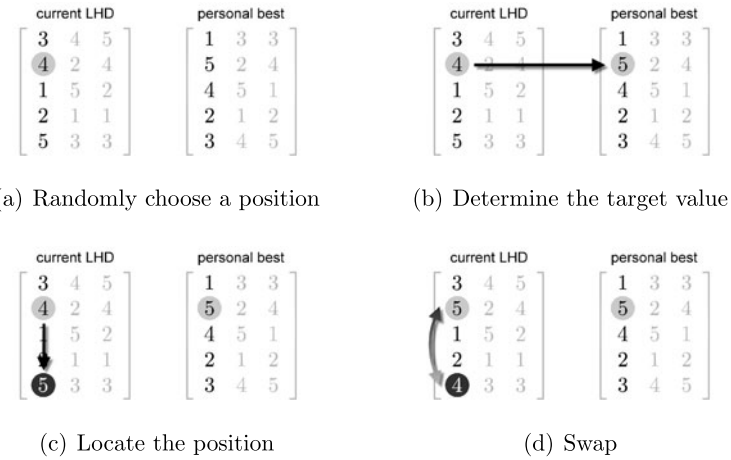


Fig. 5 Move function. (**a**) Randomly choose a position in a *column* of the current LHD. (**b**) Determine the target value in the corresponding personal best position. (**c**) Search in the current LHD *column* and locate the position possessing the target value. (**d**) Swap the values in the two positions

are specified. In addition, given a specified permutation of the elements, the LaPSO algorithm leads to LHDs that have same or very close ϕ_p values in different LaPSO runs. The corresponding designs may or may not be the same.

4 Numerical results

In Sects. 4.1 and 4.2, we examine the effects of *SameNumP* and *SameNumG*. As shown in Sect. 4.2, the results lead to three particular types of LaPSO: LaPSO-P (*SameNumG* $= 0$), LaPSO-G (*SameNumP* $= 0$), and LaPSO-L (with multiple particle groups). Representative numerical results of these three types of LaPSO are analyzed to characterize their properties. Then we compare LaPSO with two popular search algorithms: the genetic algorithm (GA; Liefvendahl and Stocki 2006) and the enhanced stochastic evolutionary algorithm (ESE; Jin et al. 2005) in Sects. 4.3 and 4.4, respectively. Finally, we present the best extended max-min LHDs found by using LaPSO with the ϕ_p criterion in Sect. 4.5. Some details of the experiments are as follows. The experiments in Sects. 4.1, 4.2, 4.3, and 4.4 are all repeated 1,000 times. The ϕ_p value is computed after scaling the LHD into $[0, 1]^k$, and we use the Euclidean norm for inter-site distances. Except in Sect. 4.5, the value of p in ϕ_p criterion is set at 50, as suggested in Jin et al. (2005), Viana et al. (2010), Ye et al. (2000).

We accelerate LaPSO by using an NVIDIA Tesla C2070 GPU to learn the behaviors of LaPSO from more experimental results and search for the optimal LHDs in a shorter time. The main structure of the implementation and acceleration schemes are similar to Hung and Wang (2012). We let one GPU thread take charge of one particle (or one LHD) and launch a separate CUDA kernel for each step to achieve necessary synchronization. The unchanging parameters, such as n and k, reside in the constant memory. To have coalesced global memory accesses, the LHDs are stored as shown in Fig. 6. Table 2 compares the timing of the GPU version with the timing of the CPU version (both in seconds) that

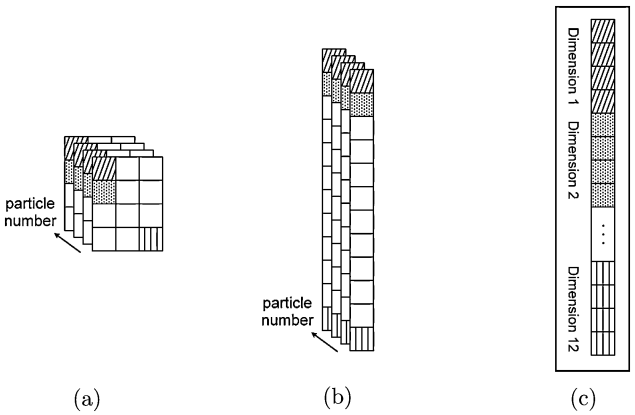


Fig. 6 (**a**) Conceptual structure of LHDs. (**b**) Vectorization of each LHD along *columns*. (**c**) The arrangement in the device global memory

Table 2 Timing results in seconds for the CPU and GPU versions of LaPSO for $(n, k) = (7, 20)$. The number of iterations is 100, 500, and 1,000

Iterations	Particle number	CPU ver.	GPU ver.	Speedup
100	10,240	3.3629	0.1932	17×
	51,200	16.8155	0.7334	23×
	102,400	33.6364	1.4143	24×
	512,000	168.1636	6.5478	26×
500	10,240	16.3057	0.6529	25×
	51,200	82.0092	2.0674	40×
	102,400	163.9962	3.7952	43×
	512,000	819.8735	17.1088	48×
1,000	10,240	33.8559	1.1098	31×
	51,200	171.2104	3.5908	48×
	102,400	342.4037	6.6562	51×
	512,000	1763.0860	29.9081	59×

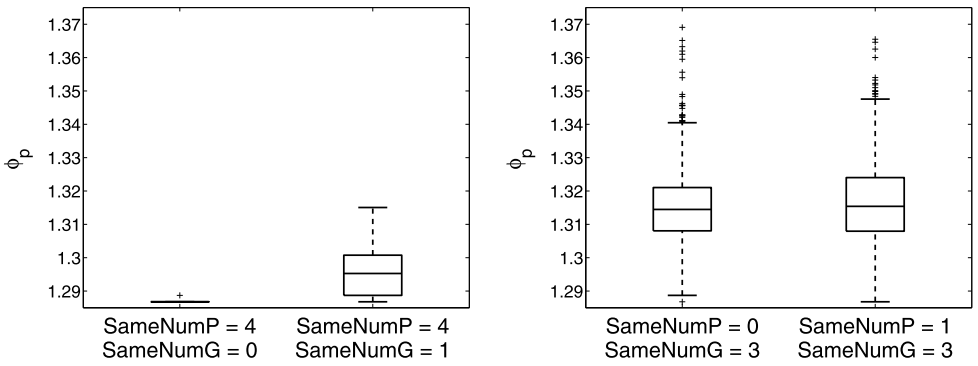


Fig. 7 The effects of both non-zeros *SameNumP* and *SameNumG*. The y-axis represents ϕ_p with $p = 50$

(a) The effect of $SameNumG$. (b) The effect of $SameNumP$.

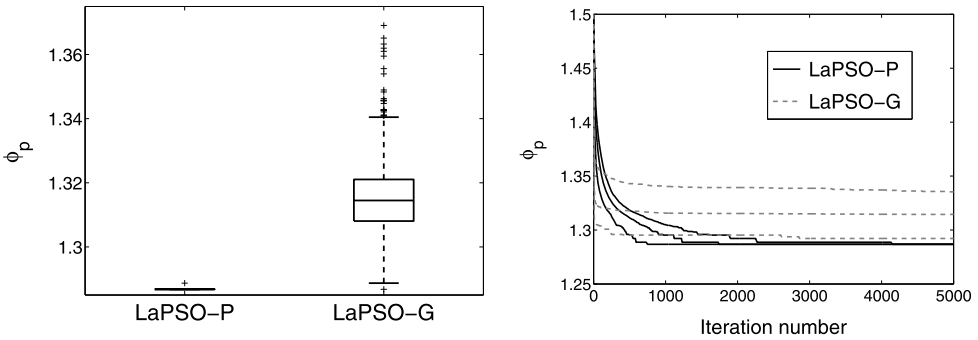


Fig. 8 Behaviors of LaPSO-P and LaPSO-G. The y-axis represents the ϕ_p value with $p = 50$. For (**b**), the x-axis represents the number of iterations. The three lines of the same style in (**b**) from bottom up are the 5th, 50th, and 95th percentiles of the ϕ_p values of the global best LHDs at each iteration over 1,000 repetitions

(a) (b)

is run on a HP BL460cG6 workstation equipped with Intel Xeon X5570 2.93 GHz CPU. The table suggests that the GPU implementation saves significant computational time, especially for larger numbers of particles and iterations.

4.1 Effects of *SameNumP* and *SameNumG*

Following the idea of the continuous PSO, it is tempting to set *SameNumP* $\neq 0$ and *SameNumG* $\neq 0$ in LaPSO. We examine the relationship between ϕ_p and the Hamming distance (Fig. 4) to see that this is not a good approach. As we have observed, LHDs are more likely to have similar ϕ_p if the LHDs are close in the sense of Hamming distance. Suppose we are now given a LHD of medium quality. For LHDs that are far from this LHD, we expect large differences in ϕ_p values, and almost all ϕ_p values become larger because of the great portion of bad LHDs. On the other hand, we anticipate that within LHDs that are close to the given LHD, there will be a better LHD with greater chance. In view of the above observation, it is better to concentrate the search in the neighborhood of *one* LHD. If both *SameNumP* and *SameNumG* are not zero, the attractions toward personal and global bests are actually distractions from the search in the neighborhood of global best or personal best. The results of 9-run 4-factor LHDs, shown in Fig. 7, demonstrate this idea. We see the effect of *SameNumG* in Fig. 7(a) and the effect of *SameNumP* in Fig. 7(b). *SameNumG* $= 1$ yields poor results in Fig. 7(a). Because we first move toward personal best and then toward global best, the results suggest that the moving toward global best distracts particles from better positions. On the other hand, *SameNumP* $= 1$ in Fig. 7(b) has only a mild effect on the results, but this figure additionally indicates that *SameNumP* $= 1$ dost not lead to an improvement.

4.2 Characteristics of LaPSO in three types

Based on the observation in Sect. 4.1, we set one of *SameNumP* and *SameNumG* to 0 hereafter. We call "*SameNumP* $\neq 0$ and *SameNumG* $= 0$" LaPSO-P, which focuses on neighborhoods of personal bests, and "*SameNumP* $= 0$ and *SameNumG* $\neq 0$" LaPSO-G, which concentrates the search on the neighborhood of global best. Below we examine behaviors of LaPSO in the two settings.

Figure 8(a) contains the results of LaPSO-P and LaPSO-G from Fig. 7(a) (*SameNumP* $= 4$ and *SameNumG* $= 0$) and Fig. 7(b) (*SameNumP* $= 0$ and *SameNumG* $= 3$). In Fig. 8(a), the results of LaPSO-P are much stable than LaPSO-G. However, if we examine the plot of ϕ_p value versus the iteration in Fig. 8(b), the stability comes at the expense of a slower decrease in the ϕ_p value. There is one more lesson from Fig. 8(a). Upon further examination of the 1,000 LHDs given by LaPSO-G, there are 988 distinct

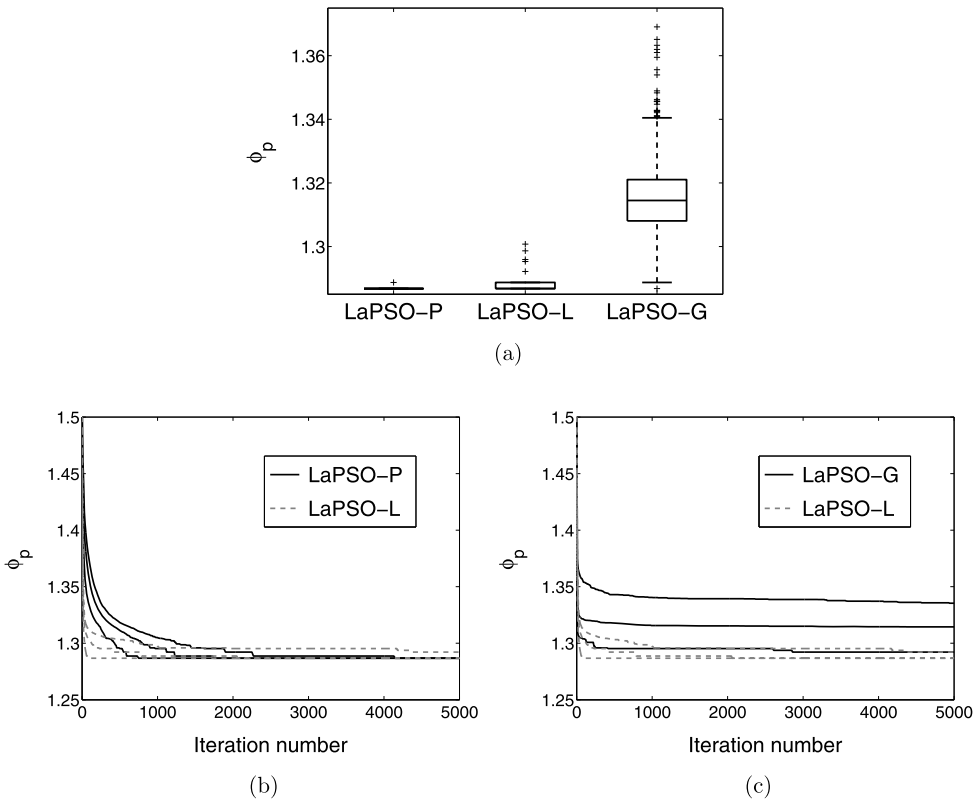


Fig. 9 Behaviors of LaPSO-P, LaPSO-L, and LaPSO-G. The y-axis represents the ϕ_p value with $p = 50$. For (**b**) and (**c**), the x-axis represents the number of iterations. *The three lines* of the same style in (**b**) and (**c**) from *bottom up* are the 5th, 50th, and 95th percentiles of the ϕ_p values of the global best LHDs at each iteration over 1,000 repetitions

LHDs, and all of them are local minima in the sense of Hamming distance, that is, performing any swap in the LHD will increase the ϕ_p value. This finding indicates that there are many local minima in this problem and additionally explains the behaviors of LaPSO-P and LaPSO-G. For LaPSO-G, all particles search the neighborhood of the global best, so the function value drops rapidly. However, because all particles focus on only one area, they are easily trapped in a local minimum. On the other hand, each particle of LaPSO-P searches the neighborhood of its personal best. Because each particle searches independently, the function value decreases more slowly than LaPSO-G. All particles of LaPSO-P may eventually be local minima, but the result is usually better because particles in LaPSO-P explore more space than particles in LaPSO-G.

The results from LaPSO-P are better than LaPSO-G, provided that the iteration is sufficiently long. LaPSO-P usually needs five times more iterations than LaPSO-G. Because the slow decreasing of function value in LaPSO-P is due to the fact that each particle searches independently, we use slightly more particles to help the search. To be more precise, we divide all the particles into several groups based on particle indexes; then each particle moves toward the best position ever visited by the group, which is called the local best. This type of LaPSO will be called LaPSO-L. LaPSO-P and LaPSO-G are special cases of LaPSO-L with group size one and all particles. Figure 9 shows the behaviors of LaPSO-L with the group size 128. From Fig. 9(a), we lose some stability compared to LaPSO-P, but the decreasing speed now becomes faster and is similar to LaPSO-G as shown in Figs. 9(b) and 9(c). Although we present only one case here, the properties we mention above can be found in other cases. All parameters used in Figs. 7, 8, and 9 are listed in Table 3.

Some general suggestions about the parameters are as follows. Usually, the performance of LaPSO is quite good when $SameNumP$ is approximately $n/2$ for LaPSO-P and $SameNumG$ and $SameNumL$ are approximately $n/4$ for LaPSO-G and LaPSO-L. As for the random probability $ProbR$, we do not want to over perturb the LHDs, so $ProbR \times (k-1) = 1$ to 2 is often sufficient.

4.3 Comparison with GA

The GA algorithm by Liefvendahl and Stocki (2006) uses an elite strategy that is similar to LaPSO-G, concentrating on the neighborhood of the global best, so we first compare GA to LaPSO-G. Table 4 shows the results of LaPSO-G and GA in 6 cases. In these cases, the results from both methods have similar trends, which is additionally confirmed by the results in Figs. 10(a) and 10(b). We also present the results of LaPSO-L in Table 4. Unlike LaPSO-G and GA, which focus the search near the global best, LaPSO-L can explore more space in different particle groups, which makes the results of LaPSO-L more stable than LaPSO-G and GA. This stability

Table 3 The parameters used in Figs. 7(a), 7(b), 8, and 9

n	k	p	Method	Particle No.	Iteration No.	*SameNumP*	*SameNumG*	*ProbR*
The parameters to Fig. 7(a)								
9	4	50	LaPSO	51,200	5,000	4	0	0.5
						4	1	
The parameters to Fig. 7(b)								
9	4	50	LaPSO	51,200	5,000	0	3	0.5
						1	3	

n	k	p	Method	Particle No.	Iteration No.	*SameNum*	*ProbR*
The parameters to Fig. 8							
9	4	50	LaPSO-P	51,200	5,000	4	0.5
			LaPSO-G			3	

n	k	p	Method	Particle No.	Iteration No.	Group size	*SameNum*	*ProbR*
The parameters to Fig. 9								
9	4	50	LaPSO-P	51,200	5,000	–	4	0.5
			LaPSO-L			128	3	
			LaPSO-G			–	3	

can be observed from Table 4; many of the designs found by LaPSO-L have a smaller ϕ_p than the other two methods.

4.4 Comparison with ESE

ESE proposed by Jin et al. (2005) adopts a different idea than LaPSO-L to address local minima. LaPSO-L tries to cover local minima using many particles, while ESE tries to move the current LHD from one local minimum to another local minimum. In Table 5, LaPSO-L and LaPSO-G are compared with ESE. The ESE outperforms LaPSO-G in the simulations, which is not surprising, given the fact that ESE searches more local minima than LaPSO-G. The table shows that the chance of finding a good design using LaPSO-L is higher than ESE, suggesting that the numbers of particles we use are sufficient to cover local minima in these cases. If we use enough particles, LaPSO-L can obtain better results, even for cases with greater k, as shown in Table 6. However, for cases with large n, LaPSO-L may not perform better than ESE. Because LaPSO-L has little chance to escape from local minima, the performance of LaPSO-L is affected by whether the local minima are covered by the particles. Currently, we increase the coverage of the space by using more particles, but this approach has its limit, especially when the problem becomes large. Increasing the number of particles may not increase the coverage efficiently, and we cannot increase the number of particles unboundedly due to memory constraints. In practice, to improve the performance and efficiency of LaPSO-L, we suggest trying to increase the coverage of the space or adding a mechanism to LaPSO-L such that it can move away from local minima.

4.5 A collection of optimal max-min designs

In this section, we show how we search for an optimal extended max-min LHD using the ϕ_p criterion. To decide the p value, we monitor the minimum pairwise distance of the global best LHD during the iterations of LaPSO. If the minimum distance decreases during the iterations, LaPSO is restarted with p increased by 10. Repeating this process, we choose the smallest p such that the minimum pairwise distance of the global best LHD is non-decreasing during the iterations for 50 consecutive runs of LaPSO.

To assess the quality of LHDs we found, the squared minimum distances (d_1^2) of optimal LHDs generated by LaPSO and other approaches are presented in Table 7. We additionally report the number of pairs of design points (J_1) separated by the minimum distances. The column UB is the theoretical upper bounds of the max-min designs derived by van Dam et al. (2009). The column WEB is the results from http://www.spacefillingdesigns.nl, which contains max-min LHDs collected from several papers. These WEB results are not necessarily global optimal solutions. If we compare the results of LaPSO with the results from the website, LaPSO is capable of achieving the same minimal inter-site distance and even making some improvements. For others that are not available from the website, the results obtained by LaPSO are equal or reasonably close to the upper bounds.

In Table 8, we present some improved results over those found in Morris and Mitchell (1995), denoted by M&M. Table 8 shows that the LHD obtained by LaPSO has a larger minimal inter-site distance than that of M&M for the cases where $(n, k) = (9, 9)$ and $(12, 5)$. For other cases, M&M and

Table 4 A comparison of LaPSO-L, LaPSO-G, and GA (*the first part*) and the algorithm parameters (*the second part*)

$n \times k$	Method	Min	Percentiles				Max
			5	25	50	75	
8×3	LaPSO-L	1.6054*	1.6054	1.6054	1.6054	1.6054	1.6054
	LaPSO-G	1.6054*	1.6054	1.6232	1.6335	1.6335	1.6931
	GA	1.6054*	1.6054	1.6232	1.6335	1.6335	1.6955
8×4	LaPSO-L	1.1510	1.1510	1.1510	1.1510	1.1510	1.1510
	LaPSO-G	1.1510	1.1510	1.1510	1.1725	1.1725	1.2846
	GA	1.1510	1.1510	1.1510	1.1725	1.2178	1.2848
8×5	LaPSO-L	1.0329	1.0329	1.0329	1.0329	1.0370	1.0446
	LaPSO-G	1.0329	1.0394	1.0488	1.0543	1.0594	1.0865
	GA	1.0329	1.0370	1.0483	1.0538	1.0594	1.0797
10×3	LaPSO-L	1.7861	1.7861	1.7861	1.7861	1.7861	1.7861
	LaPSO-G	1.7861	1.7861	1.8151	1.8278	1.8527	1.9292
	GA	1.7861	1.7861	1.8278	1.8521	1.8582	1.9546
10×4	LaPSO-L	1.3402	1.3402	1.3513	1.3524	1.3598	1.3651
	LaPSO-G	1.3513	1.3652	1.3745	1.3828	1.3912	1.4214
	GA	1.3402	1.3643	1.3736	1.3814	1.3893	1.4160
10×5	LaPSO-L	1.0670	1.0768	1.0777	1.0788	1.0827	1.0885
	LaPSO-G	1.0670	1.0825	1.0968	1.1202	1.1440	1.1872
	GA	1.0670	1.0820	1.0915	1.1154	1.1398	1.1753

*The true minimum by exhaustive search

n	k	p	Method	Particle no.	Ite. no.	Group size	*SameNum*	*Prob**
8	3	50	LaPSO-L	10,240	1,000	32	2	0.3
			LaPSO-G			–	2	0.3
			GA			–	–	0.4
8	4	50	LaPSO-L	30,720	2,000	32	2	0.3
			LaPSO-G			–	2	0.6
			GA			–	–	0.7
8	5	50	LaPSO-L	51,200	3,000	32	2	0.3
			LaPSO-G			–	2	0.6
			GA			–	–	0.7
10	3	50	LaPSO-L	51,200	3,000	64	3	0.3
			LaPSO-G			–	3	0.7
			GA			–	–	0.7
10	4	50	LaPSO-L	204,800	4,000	64	3	0.5
			LaPSO-G			–	3	0.7
			GA			–	–	0.7
10	5	50	LaPSO-L	409,600	5,000	64	3	0.5
			LaPSO-G			–	3	0.7
			GA			–	–	0.7

**ProbR* for LaPSO-L and LaPSO-G. Probability of mutation for GA

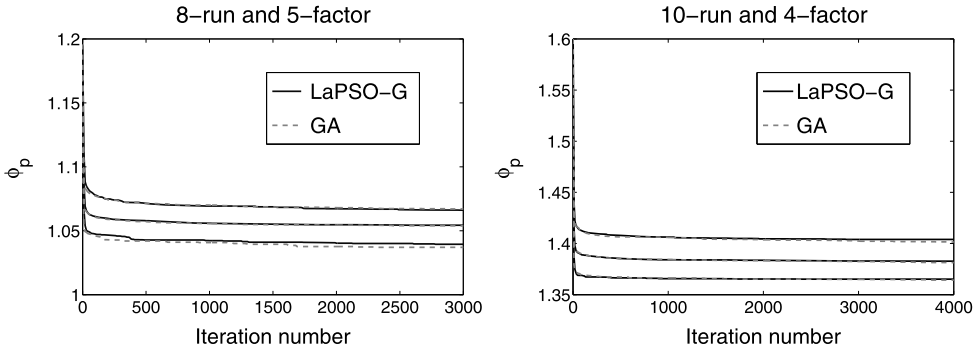


Fig. 10 ϕ_p versus the number of iterations of LaPSO-G and GA. The x- and y-axis represent the number of iterations and ϕ_p value with $p = 50$, respectively. In addition, $(n, k) = (8, 5)$ for (**a**) and $(n, k) = (10, 4)$ for (**b**). *The three lines* of the same style from bottom up are the 5th, 50th, and 95th percentiles of the ϕ_p values of the global best LHDs over 1,000 repetitions

Table 5 A comparison of LaPSO-L, LaPSO-G, and ESE. The parameters of LaPSO-L and LaPSO-G are the same as in Table 4. The parameters of ESE follow the suggestion of Jin et al. (2005)

$n \times k$	Method	Min	Percentiles				Max
			5	25	50	75	
8×3	LaPSO-L	1.6054*	1.6054	1.6054	1.6054	1.6054	1.6054
	LaPSO-G	1.6054*	1.6054	1.6232	1.6335	1.6335	1.6931
	ESE	1.6054*	1.6054	1.6054	1.6054	1.6070	1.6232
8×4	LaPSO-L	1.1510	1.1510	1.1510	1.1510	1.1510	1.1510
	LaPSO-G	1.1510	1.1510	1.1510	1.1725	1.1725	1.2846
	ESE	1.1510	1.1510	1.1510	1.1510	1.1510	1.1725
8×5	LaPSO-L	1.0329	1.0329	1.0329	1.0329	1.0370	1.0446
	LaPSO-G	1.0329	1.0394	1.0488	1.0543	1.0594	1.0865
	ESE	1.0329	1.0329	1.0370	1.0400	1.0420	1.0513
10×3	LaPSO-L	1.7861	1.7861	1.7861	1.7861	1.7861	1.7861
	LaPSO-G	1.7861	1.7861	1.8151	1.8278	1.8527	1.9292
	ESE	1.7861	1.7861	1.7861	1.7861	1.8146	1.8194
10×4	LaPSO-L	1.3402	1.3402	1.3513	1.3524	1.3598	1.3651
	LaPSO-G	1.3513	1.3652	1.3745	1.3828	1.3912	1.4214
	ESE	1.3402	1.3524	1.3622	1.3652	1.3686	1.3768
10×5	LaPSO-L	1.0670	1.0768	1.0777	1.0788	1.0827	1.0885
	LaPSO-G	1.0670	1.0825	1.0968	1.1202	1.1440	1.1872
	ESE	1.0670	1.0768	1.0820	1.0845	1.0881	1.1003

*The true minimum by exhaustive search

LaPSO share the same minimal inter-site distance. However, the LHDs found by LaPSO have fewer pairs of design points that attain the minimal inter-site distance. In other words, LaPSO performs better in terms of the extended max-min criterion in these cases.

5 Conclusion

In this paper, we propose LaPSO for optimizing LHD. Due to the relationship between the ϕ_p criterion and the Hamming distance, we focus the search on the neighborhood of one LHD and examine the derivative LaPSO-G, LaPSO-P, and LaPSO-L. For LaPSO-G, ϕ_p drops very quickly, but the results have larger variation because the particles may be trapped in a local minimum. As for LaPSO-P, ϕ_p decreased mildly compared to LaPSO-G, but the results are much more stable if the iteration is sufficiently long. LaPSO-L is a compromise between the speed of function value decreasing and the stability of the results. We additionally show how we can find the extended max-min LHD using the ϕ_p criterion. Both C and MATLAB codes of LaPSO for ϕ_p criterion are available from the authors upon request.

Possible future work includes choosing initial LHDs in LaPSO with good coverage of the space rather than random ones and devising a mechanism to make LaPSO escape

Table 6 A comparison of LaPSO-L and ESE in the cases with larger k. The algorithm parameters are listed. Parameters of ESE follow the suggestion of Jin et al. (2005)

$n \times k$	Method	Min	Percentiles				Max
			5	25	50	75	
6×20	LaPSO-L	0.4466	0.4466	0.4466	0.4466	0.4466	0.4467
	ESE	0.4466	0.4467	0.4467	0.4468	0.4468	0.4469
7×20	LaPSO-L	0.4669	0.4670	0.4670	0.4670	0.4671	0.4671
	ESE	0.4670	0.4672	0.4673	0.4674	0.4675	0.4676
8×20	LaPSO-L	0.4832	0.4833	0.4833	0.4834	0.4834	0.4835
	ESE	0.4834	0.4836	0.4837	0.4838	0.4838	0.4840

n	k	p	Method	Particle no.	Ite. no.	Group size	*SameNum*	*Prob*
6	20	50	LaPSO-L	102,400	4,000	32	2	0.1
7	20	50		512,000	4,000	32	2	0.1
8	20	50		1,024,000	4,000	64	2	0.1

Table 7 A comparison of the squared minimum distance. The columns "UB" and "WEB" contain the theoretical upper bounds (van Dam et al. 2009) and the results reported on the web http://www.spacefillingdesigns.nl, respectively. *The values in the parentheses* are the numbers of the minimum distance. The best results for each case are highlighted in *bold*

n	k	p	UB	WEB	LaPSO
6	10	130	70	**68** (3)	**68** (1)
6	11	140	77	74 (2)	**75** (5)
6	12	180	**84**	82 (2)	**84** (15)
6	13	180	91	**89** (6)	**89** (1)
6	14	230	98	N/A	**96** (5)
6	15	230	105	N/A	**104** (4)
6	16	240	112	N/A	**110** (1)
6	17	240	119	N/A	**117** (5)
6	18	270	**126**	N/A	**126** (15)
6	19	280	133	N/A	**131** (1)
6	20	300	140	N/A	**138** (5)
7	10	170	93	90 (1)	**91** (2)
7	11	210	102	N/A	**100** (2)
7	12	260	112	N/A	**110** (5)
7	13	280	121	N/A	**119** (2)
7	14	290	130	N/A	**128** (1)
7	15	310	140	N/A	**138** (3)
7	16	320	149	N/A	**148** (12)
7	17	330	158	N/A	**157** (4)
7	18	380	168	N/A	**166** (3)
7	19	430	177	N/A	**176** (12)
7	20	450	186	N/A	**185** (4)

from local minima while keep some information on the current state. The proposed LaPSO can be further extended to LHDs with higher-dimensional projection properties, by implementing orthogonal array-based LHDs (Tang 1993).

Table 8 Improved extended max-min results. *The values in the parentheses* are the numbers of the minimum distance

n	k	WEB	M&M	LaPSO
9	9	128 (2)	126 (1)	**129** (2)
12	4	**63** (9)	**63** (2)	**63 (1)**
12	5	**94** (4)	91 (1)	**94 (2)**
13	2	**13** (17)	**13** (17)	**13 (16)**
19	2	**18** (9)	**18** (9)	**18 (6)**
20	2	**18** (6)	**18** (5)	**18 (2)**

Acknowledgements The authors are grateful to the referees for their valuable and helpful comments and suggestions. The research of Hung is supported by National Science Foundation grants DMS 0905753 and CMMI 0927572. This work was supported in part by National Science Council under grant NSC 99-2118-M-006-006-MY2 (Chen) and NSC 100-2628-M-002-011-MY4 (Wang), the Taida Institute of Mathematical Sciences, Center for Advanced Study in Theoretical Sciences, the National Center for Theoretical Sciences (Taipei Office) and the Mathematics Division of the National Center for Theoretical Sciences (South) in Taiwan.

References

Bratton, D., Kennedy, J.: Defining a standard for particle swarm optimization. In: IEEE Swarm Intelligence Symposium. SIS 2007, pp. 120–127. IEEE, Piscataway (2007)

Bates, S.J., Sienz, J., Toropov, V.V.: Formulation of the optimal Latin hypercube design of experiments using a permutation genetic algorithm. In: AIAA 2004-2011, pp. 1–7 (2004)

Engelbrecht, A.P.: Fundamentals of Computational Swarm Intelligence. Wiley, Hoboken (2006)

Fang, K.-T., Ma, C.-X., Winker, P.: Centered L2-discrepancy of random sampling and Latin hypercube design, and construction of uniform designs. Math. Comput. **71**, 275–296 (2002)

Grosso, A., Jamali, A.R.M.J.U., Locatelli, M.: Finding maximin Latin hypercube designs by iterated local search heuristics. Eur. J. Oper. Res. **197**(2), 541–547 (2009)

Hung, Y., Wang, W.: Accelerating parallel particle swarm optimization via GPU. Optim. Methods Softw. **27**(1), 33–51 (2012)

Jin, R., Chen, W., Sudjianto, A.: An efficient algorithm for constructing optimal design of computer experiments. J. Stat. Plan. Inference **134**(1), 268–287 (2005)

Joseph, V.R., Hung, Y.: Orthogonal-maximin Latin hypercube designs. Stat. Sin. **18**, 171–186 (2008)

Kennedy, J., Eberhart, R.C.: Particle swarm optimization. In: Proceedings of IEEE International Conference on Neural Networks, vol. 4, pp. 1942–1948. IEEE, Piscataway (1995)

Kennedy, M., O'Hagan, A.: Predicting the output from a complex computer code when fast approximations are available. Biometrika **87**, 1–13 (2000)

Liefvendahl, M., Stocki, R.: A study on algorithms for optimization of Latin hypercubes. J. Stat. Plan. Inference **136**(9), 3231–3247 (2006)

Morris, M.D., Mitchell, T.J.: Exploratory designs for computational experiments. J. Stat. Plan. Inference **43**(3), 381–402 (1995)

Santner, T.J., Williams, B.J., Notz, W.I.: The Design and Analysis of Computer Experiments. Springer, New York (2003)

Tang, B.: Orthogonal array-based Latin hypercubes. J. Am. Stat. Assoc. **88**(424), 1392–1397 (1993)

Toala, D.J.J., Bressloff, N.W., Keanea, A.J., Holden, C.M.E.: The development of a hybridized particle swarm for kriging hyperparameter tuning. Eng. Optim. **43**(6), 675–699 (2011)

van Dam, E.R., Husslage, B., den Hertog, D., Melissen, H.: Maximin Latin hypercube designs in two dimensions. Oper. Res. **55**(1), 158–169 (2007)

van Dam, E.R., Rennen, G., Husslage, B.: Bounds for maximin Latin hypercube designs. Oper. Res. **57**(3), 595–608 (2009)

van den Bergh, F.: An analysis of particle swarm optimizers. Ph.D. Thesis, University of Pretoria (2006)

van den Bergh, F., Engelbrecht, A.P.: A new locally convergent particle swarm optimiser. In: IEEE International Conference on Systems, Man and Cybernetics, p. 3 (2002)

Viana, F.A.C., Venter, G., Balabanov, V.: An algorithm for fast optimal Latin hypercube design of experiments. Int. J. Numer. Methods Eng. **82**(2), 135–156 (2010)

Ye, K.Q., Li, W., Sudjianto, A.: Algorithmic construction of optimal symmetric Latin hypercube designs. J. Stat. Plan. Inference **90**(1), 145–159 (2000)