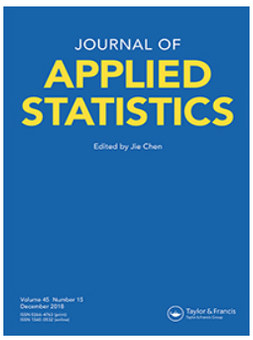

Journal of Applied Statistics

ISSN: 0266-4763 (Print) 1360-0532 (Online) Journal homepage: https://www.tandfonline.com/loi/cjas20

Optimal orthogonal-array-based latin hypercubes

Stephen Leary , Atul Bhaskar & Andy Keane

To cite this article: Stephen Leary , Atul Bhaskar & Andy Keane (2003) Optimal orthogonal-array-based latin hypercubes, Journal of Applied Statistics, 30:5, 585-598, DOI: 10.1080/0266476032000053691

To link to this article: https://doi.org/10.1080/0266476032000053691

Journal of Applied Statistics, Vol. 30, No. 5, 2003, 585–598

Routledge
Taylor & Francis Group

Optimal orthogonal-array-based latin hypercubes

STEPHEN LEARY, ATUL BHASKAR & ANDY KEANE, *Computational Engineering and Design Centre, University of Southampton, UK*

ABSTRACT *The use of optimal orthogonal array latin hypercube designs is proposed. Orthogonal arrays were proposed for constructing latin hypercube designs by Tang (1993). Such designs generally have better space filling properties than random latin hypercube designs. Even so, these designs do not necessarily fill the space particularly well. As a result, we consider orthogonal-array-based latin hypercube designs that try to achieve optimality in some sense. Optimization is performed by adapting strategies found in Morris & Mitchell (1995) and Ye* et al. *(2000). The strategies here search only orthogonal-array-based latin hypercube designs and, as a result, optimal designs are found in a more efficient fashion. The designs found are in general agreement with existing optimal designs reported elsewhere.*

1 Introduction

A major application area for designed experiments is to enable the optimization of complex engineering systems. Such systems are generally represented by expensive computer codes (for instance, a finite element solution in a structural analysis, or a Navier–Stokes solution in computational fluid dynamics (CFD)). Due to the expense of the code, direct optimization is quite often infeasible and approximation methods are then considered. The output of these expensive computer codes can be, for instance, approximated using a response surface model, see for example, Myers & Montgomery (1995), or by a kriging model, for example Jones *et al.* (1998).

Correspondence: Andy Keane, Computational Engineering and Design Centre, School of Engineering Sciences, University of Southampton, Southampton SO17 1BJ, UK

ISSN 0266-4763 print; 1360-0532 online/03/050585-14
DOI: 10.1080/0266476032000053691

One key question when the available data are to be limited due to the cost of running the model is: at which values of the input variables do we run the code? In the absence of any *a priori* information on the system of interest, it is commonly recognized that some form of uniformity of the design points throughout the region of interest is favourable. Many experimental design planning methods exist in the literature and an exhaustive list is well beyond the scope of this paper. Here, we restrict ourselves to latin hypercube designs, first introduced by McKay *et al.* (1979).

A latin hypercube is an n by m matrix, each column of which is a permutation of $1, 2, \ldots, n$. Such designs are seen to enjoy good 'space-filling' properties, covering the design space well without replication. However, one must note that amongst various latin hypercube designs, some are better than others, so optimal latin hypercube designs have recently been considered (Park, 1994; Tang, 1994; Morris and Mitchell, 1995; Ye, 1998 and Ye *et al.*, 2000).

A second question one may ask is: how do we assess such designs? Previous approaches to experimental designs have included the use of maximum entropy designs (Shewry & Wynn, 1987; Currin *et al.*, 1991), integrated mean squared error of prediction (Sacks *et al.*, 1989) and minimax and maximin distance designs (Johnson *et al.*, 1990).

Here we consider an alternative distance related metric

$$\sum_{i=1}^{n} \sum_{j=i+1}^{n} \frac{1}{d_{ij}^2} \tag{1}$$

where n is the number of sampled points and d_{ij} is the Euclidean distance between points i and j. We seek to minimize this quantity in an attempt to spread the design points as uniformly as possible. This criterion is most closely related to maximin designs. Experience suggests (Ye *et al.*, 2000) that the resulting approximations are relatively insensitive to optimal design criteria. The designs found herein are also in agreement with those presented elsewhere (see for instance, Park, 1994; Tang, 1994), despite the fact that we use a different measure. Therefore, no direct comparison will be made between different design criteria. The approach presented here is entirely general and could be implemented using any of the aforementioned criteria.

The purpose of this paper is to apply optimization strategies to search a restricted subspace of the set of all latin hypercube designs, namely orthogonal-array-based latin hypercubes. The idea is that restricting the search space should allow good designs to be found more efficiently. Previous optimization strategies for latin hypercube designs are drawn upon and we show how these algorithms can be modified to search orthogonal-array-based latin hypercube designs.

The rest of the paper is organized as follows: Section 2 briefly reviews orthogonal array latin hypercube designs and includes some simple examples to demonstrate the idea. Section 3 considers optimal designs within this class. In particular, we focus on adapting and applying the approaches introduced by Morris & Mitchell (1995), based on simulated annealing and the columnwise pairwise (CP) algorithm introduced in Li & Wu (1997), and which have been used for generating optimal symmetric latin hypercube designs in Ye *et al.* (2000). These algorithms are seen to perform better than earlier algorithms suggested in the literature (e.g. Park, 1994). Section 4 shows some results, Section 5 compares the approach to other work on optimal orthogonal array latin hypercube designs and, in Section 6, some final conclusions are drawn.

2 Orthogonal-array-based latin hypercubes

First, a brief description of orthogonal arrays is given. The approach described here follows the work of Tang (1993). An n by m matrix $\mathbf{A}$ with entries from the set $\{1, 2, \ldots, s\}$ is called an orthogonal array of strength r, size n with m constraints and s levels if each $n \times r$ submatrix of $\mathbf{A}$ contains all possible $1 \times r$ rows with the same frequency λ. Here λ is termed the index of the array, and $n = \lambda s^r$. The array is denoted by $OA(n, m, s, r)$. Here, as in Tang (1993), the s symbols are taken as $1, 2, \ldots, s$ throughout this article.

The construction of an orthogonal-array-based latin hypercube from $\mathbf{A}$ is as follows: for each column of $\mathbf{A}$, the λs^{r-1} positions with entry k are replaced by a permutation of

$$[(k-1)\lambda s^{r-1} + 1, (k-1)\lambda s^{r-1} + 2, \ldots, (k-1)\lambda s^{r-1} + \lambda s^{r-1} = k\lambda s^{r-1}] \qquad (2)$$

for all $k = 1, 2, \ldots, s$. After this is carried out for every column of $\mathbf{A}$, the newly created matrix is a latin hypercube.

For example, using $OA(9, 2, 3, 2)$ of index $\lambda = 1$ we might have

$$OA = \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 2 & 1 \\ 2 & 2 \\ 2 & 3 \\ 3 & 1 \\ 3 & 2 \\ 3 & 3 \end{bmatrix}, \qquad LH = \begin{bmatrix} 1 & 1 \\ 3 & 4 \\ 2 & 7 \\ 5 & 2 \\ 4 & 5 \\ 6 & 8 \\ 7 & 3 \\ 8 & 6 \\ 9 & 9 \end{bmatrix}$$

This design is shown in Fig. 1(a). Such designs, in general, enjoy better space filling properties than general latin hypercubes; placing the designs all on the leading diagonal still gives a latin hypercube.

Examples of latin hypercubes generated from $OA(8, 2, 2, 2)$, $OA(25, 2, 5, 2)$,

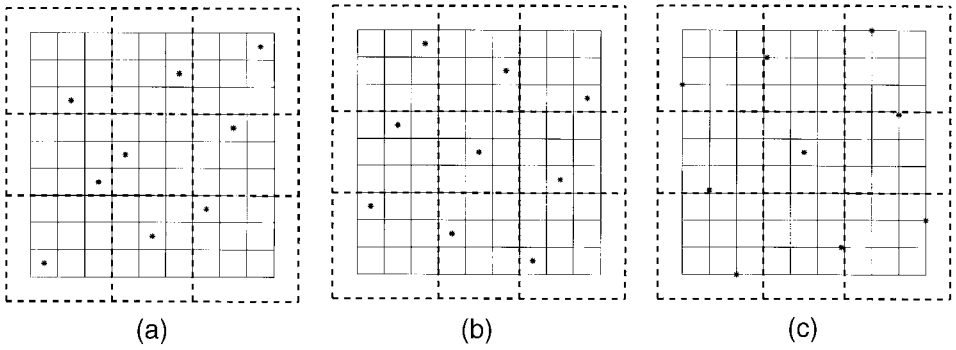

FIG. 1. (a) Random, (b) optimal midpoint and (c) optimal orthogonal-array latin hypercube design in 2D —9 points.

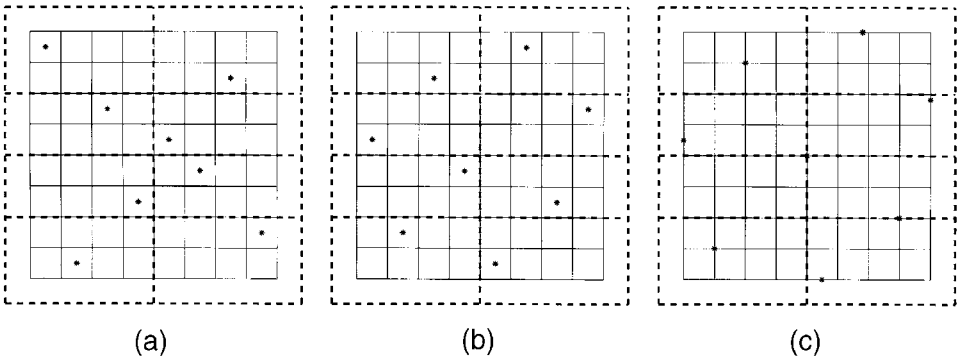

FIG. 2. (a) Random, (b) optimal midpoint and (c) optimal orthogonal-array latin hypercube design in 2D —8 points.

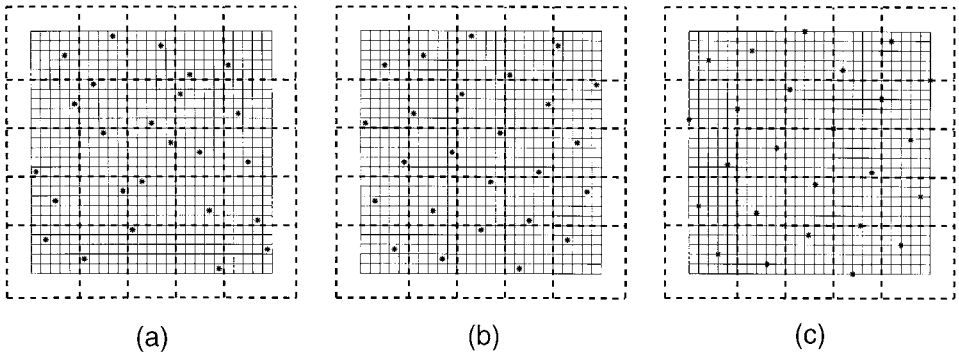

FIG. 3. (a) Random, (b) optimal midpoint and (c) optimal orthogonal-array latin hypercube design in 2D —25 points.

$OA(81, 4, 3, 4)$ and $OA(128, 7, 2, 7)$ are shown in Figs 2(a), 3(a), 4(a) and 5(a) respectively. The latter two figures show that such designs are not guaranteed to be good space filling designs. Orthogonal-array-based latin hypercubes are simply better space filling designs, on average, than general latin hypercube designs. This fact is evident from columns three and four of Table 1, which lists the results of applying equation (1) to several random 'midpoint' latin hypercube designs (see Park (1994) for a description of midpoint designs). The domain in m dimensions is taken to be $[0, 1]^m$.

The fact that random orthogonal-array-based latin hypercube designs are not as good at space filling as well-chosen latin hypercube designs (c.f. Figs 4(a), 5(a) and column four of Table 1 with Figs 4(b), 5(b) and columns five and six of Table 1 (where the latter figures and columns are generated by our algorithm described in the next section)) leads us to consider optimal orthogonal-array-based designs. (Note that it is possible to improve significantly upon latin hypercube designs if equation (1) is the only criterion being applied—for example if $m = 2$ and $n = 8$ then placing four points on the corners of the unit square and four

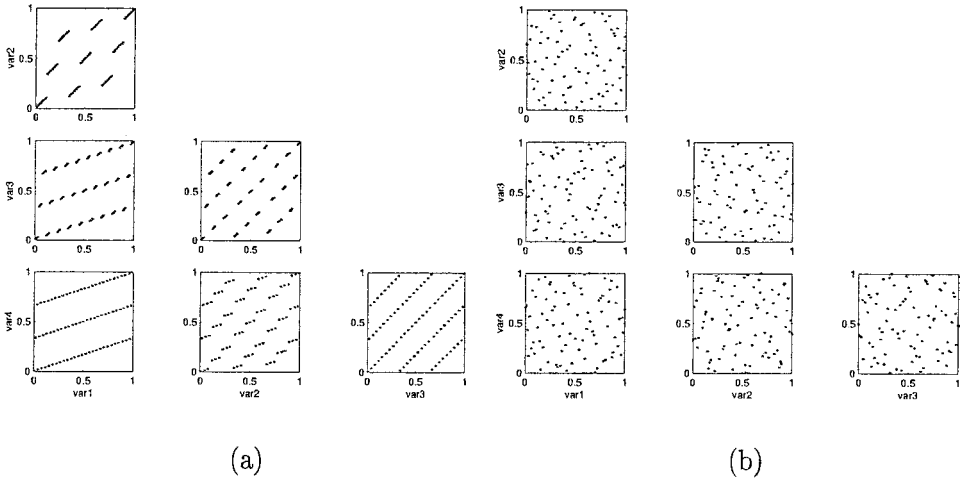

FIG. 4. (a) Random and (b) optimal orthogonal-array latin hypercube design in 4D—81 points.

midway along each side minimizes the metric at 53.4; such a strategy, however, involves a larger search space than restricting ourselves to latin hypercube designs.)

3 Optimal orthogonal-array-based latin hypercubes

In this section we adapt the approaches of Morris & Mitchell (1995) and of Ye *et al.* (2000) to search for optimal midpoint orthogonal-array-based latin hypercube designs. Both of these algorithms consider interchanging two elements within a column of the latin hypercube when searching for good candidate designs. The latter algorithm is adapted to search for symmetric designs; here we will search for optimal orthogonal-array-based latin hypercube designs. To do this, the referenced algorithms must be modified. This section gives details of the necessary modifications.

Let us consider the strategy via an example: consider the same orthogonal array ($OA(9, 2, 3, 2)$) and latin hypercube design presented earlier, and let us consider swapping, without loss of generality, the first element on the first column of the latin hypercube, see Fig. 6.

The algorithm notes this element of the orthogonal array: the entry is 1. Then we search this column of the orthogonal array, and label rows whose elements match this (here rows 2 and 3). This then provides us with possible elements that may be swapped with the initial element during optimization. In this way the orthogonal array structure of the latin hypercube is maintained. Thus, in this case, the first and third rows might be swapped, leading to the revised design of

$$LH = \begin{bmatrix} 2 & 1 \\ 3 & 4 \\ 1 & 7 \\ 5 & 2 \\ 4 & 5 \\ 6 & 8 \\ 7 & 3 \\ 8 & 6 \\ 9 & 9 \end{bmatrix}$$

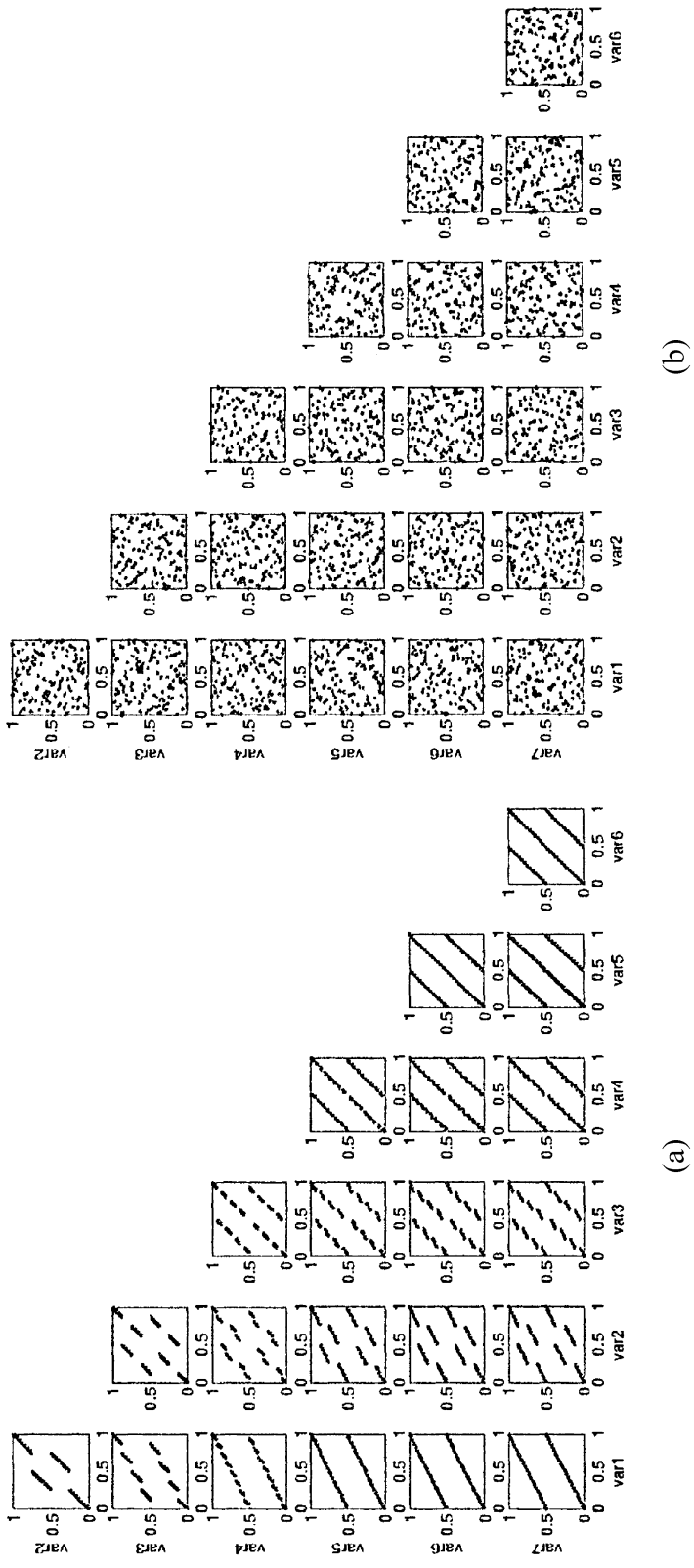

FIG. 5. (a) Random and (b) optimal orthogonal-array latin hypercube design in 7D–128 points.

TABLE 1. Value of distance metric (1) for random (averaged over 10 designs) latin hypercubes (LH) and orthogonal-array latin hypercubes (oaLH) designs, optimal midpoint oaLH designs and optimal oaLH designs of size $n \times m$

m	n	Random LH	Random oaLH	Optimum midpoint oaLH	Optimum oaLH
2	8	171.59	155.62	115.43	83.55
2	9	244.17	184.76	156.77	116.99
2	25	3311.32	2813.56	2035.79	1837.46
4	81	8644.85	8025.52	7047.16	6801.80
7	128	8971.27	8844.32	8170.79	7983.85

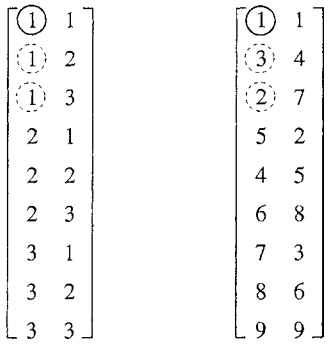

FIG. 6. Illustrative example.

(Note: this reduces the size of the search space compared with the approaches of Morris & Mitchell, 1995, and the conventional CP algorithm of Li & Wu, 1997, which both allow *any* two elements within a randomly selected column to be switched—as a result, optimal designs should be found in a more efficient manner.)

Finally, we note that these algorithms produce 'optimal' midpoint orthogonal-array latin hypercube designs. Following Park, once such a design is generated, it is optimally 'released' to find an optimal orthogonal array latin hypercube design, i.e. points are allowed to move randomly away from the midpoints of each cell. A constrained BFGS routine (Zhu *et al.*, 1994) is used for generating these final designs.

3.1 Simulated annealing algorithm for orthogonal array latin hypercube search

In the appendix, a simulated annealing algorithm is outlined to search for improved orthogonal-array-based latin hypercube designs. This algorithm is an adaptation of the algorithm proposed by Morris & Mitchell (1995). For a more detailed description of the algorithm in the case of optimizing general latin hypercube designs, we refer the reader to Morris & Mitchell (1995).

3.2 The columnwise pairwise (CP) algorithm

We now briefly describe the adapted columnwise pairwise algorithm for searching orthogonal array latin hypercube designs. Following Ye *et al.* (2000) the algorithm may be described as follows.

Step 1. Start with a random orthogonal array latin hypercube design.
Step 2. Each iteration has m steps. At the ith step, the best exchange (with respect to ϕ) of points in column i is found such that the structure of the orthogonal array latin hypercube is retained. The design matrix is updated accordingly.
Step 3. If the design is better with respect to the give criterion ϕ, repeat Step 2. Otherwise it is considered an optimal design and the search is terminated.

Essentially, this algorithm searches through each column and swaps two points in the column such that the orthogonal array structure is kept and ϕ is reduced as much as possible.

Clearly, this is a greedy downhill search and the final design is very much dependent on the initial design. Therefore, the algorithm should be started from multiple points and the best design from all these points should be considered as the final design.

4 Results

We now consider optimizing the class of designs described in Section 2. First, the optimum midpoint designs are created using the algorithms described in Section 3. The results are shown in column five of Table 1.

By comparing the results with column four of Table 1, it can be seen that considerable improvements in the designs can be achieved by making use of the optimization strategies. In the case of $m = 2, n = 8$ and $m = 2, n = 9$ these results were confirmed as the global minimum by an exhaustive search.

Some examples of optimal midpoint designs are shown in Figs 1(b), 2(b) and 3(b).

It is also worth remarking on the merits of the particular search strategies. For comparison purposes, simulated annealing (using 10 random starts) is compared with the CP algorithm (using 100 random starts). We assess the performance of each algorithm based on the convergence histories of each optimization strategy. These convergence histories plot the number of trial designs considered (x-axis) against the best current design found in terms of our distance criteria (y-axis). Brief convergence histories for two of the designs $m = 2, n = 9$ and $m = 2, n = 25$ are given in Figs 7 and 8. Here we consider the average of the ten simulated annealing runs. When considering the CP algorithm, 100 CP runs are performed and the best overall design is stored. We show only the first few thousand iterations here.

We note that, in both cases, restricting the size of the search space increases the convergence rate. The CP algorithm typically finds good solutions quickly but further improvements depend upon the initial design being considered. On average though, we expect that restricting the search to initially 'fitter' solutions ought to be beneficial.

As the problem size increases, the SA algorithm generally finds better solutions than the CP algorithm. Once again, the orthogonal-array-based approach leads to quicker convergence and better solutions, on average, when using simulated annealing. Here, the final CP design without using an orthogonal array is actually better than the CP design using orthogonal arrays, however this will not always be the case and we note the orthogonal array approach finds 'good' designs more quickly. These results are in general agreement with those reported in Ye *et al.* (2000) (where the search space was restricted to symmetric designs).

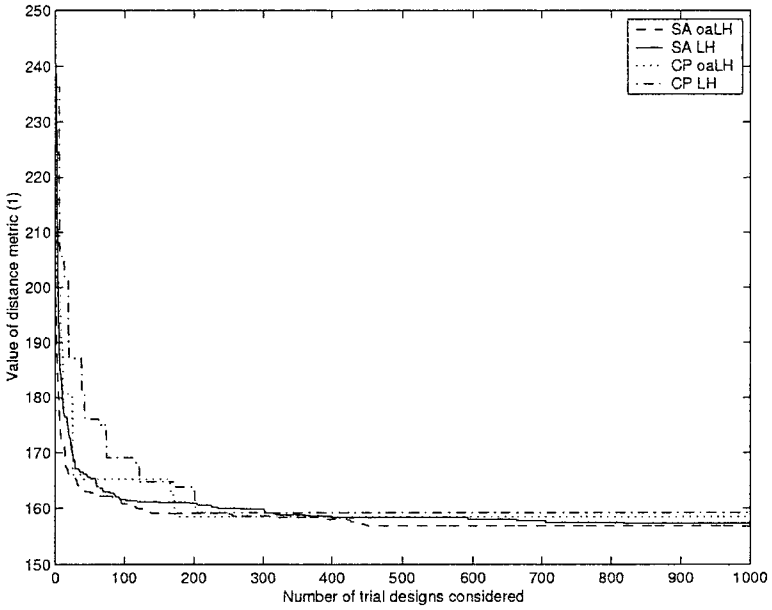

FIG. 7. Convergence histories for $m = 2, n = 9$.

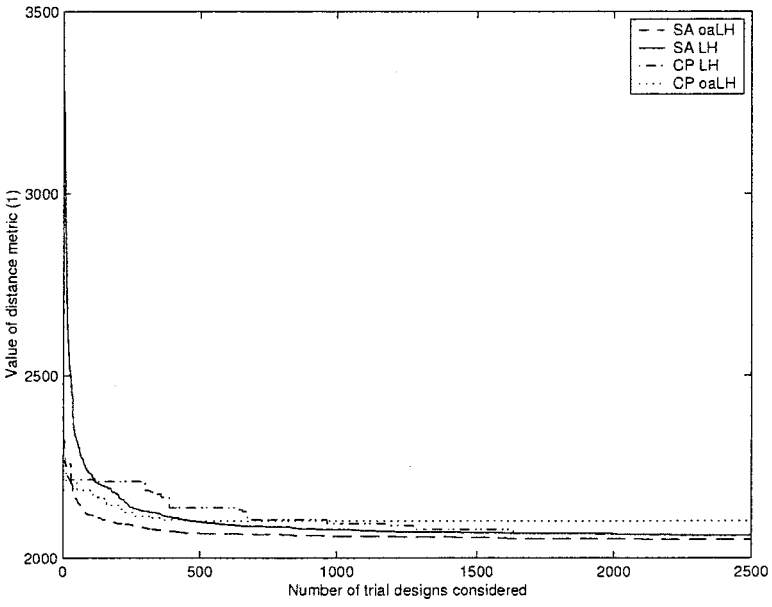

FIG. 8. Convergence histories for $m = 2, n = 25$.

We further consider an explicit timing comparison of our approach together with the standard algorithms that do not impose an orthogonal array structure on the latin hypercube design. Two examples are considered, firstly $m = 7$ and $n = 128$ and secondly $m = 16$ and $n = 256$. In both cases average convergence histories over

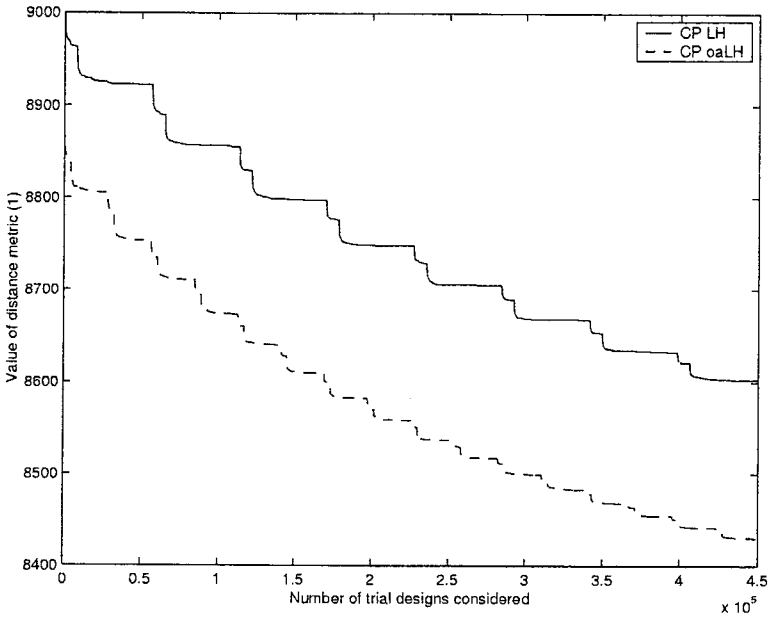

FIG. 9. Average convergence of CP algorithm for $m = 7, n = 128$.

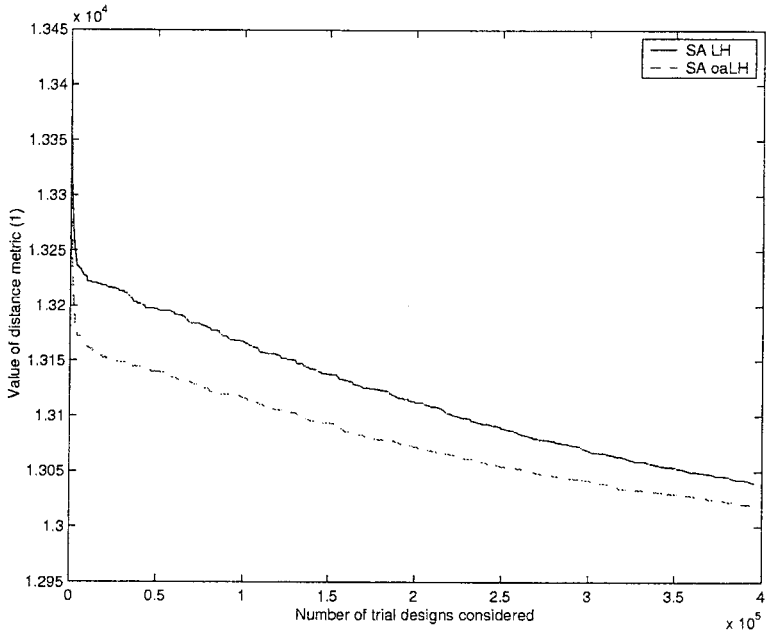

FIG. 10. Average convergence of SA algorithm for $m = 16, n = 256$.

ten optimization runs are given, these are shown in Figs 9 and 10. For $m = 7$ and $n = 128$ we used the CP algorithm and for $m = 16$ and $n = 256$ the SA algorithm is considered. Tables 2 and 3 catalogue the best result after a given number of function evaluations and the time taken to reach a target objective for these two

TABLE 2. CP optimization ($m = 7, n = 128$). Stopping criteria is when (a) MAX function evaluations are reached or (b) TARGET objective is reached (to nearest 500 runs)

MAX	Best design LH	Best design oaLH	TARGET	No. evals LH	No. evals oaLH
100 000	8855.5	8674.1	8800	132 500	28 500
200 000	8748.5	8573.4	8700	285 000	85 000
300 000	8668.5	8499.7	8600	>450 000	171 000
400 000	8662.2	8444.2	8500	>450 000	297 500
	(a)			(b)	

TABLE 3. SA optimization ($m = 16, n = 256$). Stopping criteria is when (a) MAX function evaluations are reached or (b) TARGET objective is reached (to nearest 500 runs)

MAX	Best design LH	Best design oaLH	TARGET	No. evals LH	No. evals oaLH
100 000	13 167.6	13 116.3	13 300	1 500	1 000
200 000	13 112.6	13 071.8	13 200	43 500	2 500
300 000	13 068.9	13 041.7	13 100	223 500	135 000
395 500	13 039.7	13 019.7	13 050	358 000	268 500
	(a)			(b)	

examples. Tables 2(a) and 3(a) demonstrate that if a design is to be found in a given time, then utilizing orthogonal arrays allows better designs to be found. Tables 2(b) and 3(b) show that if we specify some target of our space filling metric as a goal, then orthogonal arrays allow us to achieve this goal in less time than the standard approaches. With the CP algorithm in particular, these savings can be considerable.

We also comment on the fact that, with the simulated annealing algorithm, when considering the orthogonal-array-based approach, although we sometimes find the overall reduction in the number of function evaluations disappointing, it does, on average, lead to better designs in all cases considered when compared with a general latin hypercube search. In addition, since the design space is restricted to, on average, 'fitter' solutions, it may also make sense to start from a lower temperature. In fact, we note that different choices of the parameters of the simulated annealing algorithm could all be considered when restricting the search space. In the results presented here, the parameters were fixed for both general and orthogonal-array-based searches. However, these parameters could themselves be viewed as optimization variables and it would be interesting future work to see if a judicious choice could then be made that allowed comparable results to be achieved with even fewer function evaluations when restricting the size of the search space.

Finally, the best designs given in column five of Table 1 are optimally released to generate the final optimal orthogonal-array latin hypercube designs. The results are shown in column six of Table 1 and the final designs are shown in Figs 1(c), 2(c), 3(c), 4(b) and 5(b). Note that, in particular, the designs in Figs 4(b) and 5(b) appear to fill the space much better than the starting designs shown in Figs 4(a) and 5(a).

5 Comparison with other work on optimal orthogonal-array latin hypercube designs

As far as we are aware, this is the first paper using optimization algorithms to search orthogonal-array-based latin hypercube designs. However, in special cases, analytic results are available. An interesting special case of optimal midpoint orthogonal-array-based latin hypercube designs has been reported by Tang (1994). This is restricted to the following:

- a rectangular maximin distance criterion,
- an orthogonal array that is a single replicate factorial design.

Tang considered an $OA(s^m, m, s, m)$. Then for any $i = 1, 2, \ldots, n$ he notes that one can write in a unique way that

$$i - 1 = a_{i1} s^{m-1} + a_{i2} s^{m-2} + \ldots + a_{im} \tag{3}$$

where $a_{ij} = 0, 1, \ldots, s - 1$. He then defines

$$LH_{ij} = s^{j-1} + \sum_{p=1}^{m} a_{ip} s^{m+j-p-1} - \sum_{p=1}^{j-1} a_{ip} s^{j-p-1} \tag{4}$$

for $j = 1, \ldots, m$. Finally, he observes that LH is an optimal orthogonal-array-based latin hypercube design (it maximizes the rectangular maximin distance). This provides a benchmark result for testing the algorithms previously described. Four of the previous examples demonstrated earlier can be considered and we were able to generate globally optimal midpoint designs through our algorithmic approach. Of course, if the orthogonal array has the above properties then use of Tang's algorithm is appropriate. Nevertheless, many orthogonal array designs will not satisfy this property (e.g. $OA(8, 2, 2, 2)$ described earlier) and then our approach could be considered as a means of generating optimal designs.

6 Summary and discussion

The identification via search algorithms of optimal orthogonal-array-based latin hypercube designs under a distance-based metric has been considered. In many instances, optimal latin hypercube designs are also orthogonal-array-based designs, so a search restricted to these designs seems logical. The approach is able to reduce the time required to produce an optimal design. Further research is necessary on tuning the SA algorithm, as highlighted earlier.

The algorithms herein should be compared with other efficient strategies in the literature. We have also considered restricting the search space to symmetric latin hypercube designs as suggested in Ye *et al.* (2000). Initial results suggest our versions of the two algorithms appear to be competitive, each occasionally outperforming the other (and both always outperforming a search over all latin hypercube designs). We observe that only a simple alteration to our algorithm needs be made in order to search symmetric orthogonal-array-based designs (see Ye *et al.*, 2000, for a description of symmetric latin hypercube designs) and we note that this strategy appears consistently to outperform either of the above strategies alone.

The resulting designs have been used for selecting the input values of a finite element model of an aero-engine component and, when the resulting response was approximated using a kriging model, it generally led to a reduction in the mean

squared error (MSE) when compared with random latin hypercube or random orthogonal-array latin hypercube designs. A similar result is reported in Ye *et al.* (2000).

Of course, one restriction with our approach is that we must consider latin hypercube designs for which an orthogonal array exists. When this is the case, our approach quickly leads to very good space filling designs.

Acknowledgements

We would like to thank Rolls-Royce plc for the financial support received as a part of the University Technology Partnership (UTP) in design. Finally, we would like to thank Professor Sue Lewis of the Statistics Group, Faculty of Mathematical Studies, University of Southampton for her useful suggestions on an earlier version of this manuscript.

REFERENCES

Currin, C., Mitchell, T., Morris, M. D. & Ylvisaker, D. (1991) Bayesian prediction of deterministic functions, with applications to the design and analysis of computer experiments, *Journal of the Americal Statistical Association*, 86, pp. 953–963.

Johnson, M., Moore, L. & Ylvisaker, D. (1990) Minimax and maximin distance designs, *Journal of Statistical Planning and Inference*, 26, pp. 131–148.

Jones, D. R., Schonlau, M. & Welch, W. J. (1998) Efficient global optimization of expensive black-box functions, *Journal of Global Optimization*, 13(4), pp. 455–492.

Li, W. & Wu, C. F. J. (1997) Columnwise-pairwise algorithms with applications to the construction of supersaturated designs, *Technometrics*, 39, pp. 171–179.

McKay, M. D., Beckman, R. J. & Conover, W. J. (1979) A comparison of three methods for selecting values of input variables in the analysis of output from a computer code, *Technometrics*, 21, pp. 239–245.

Morris, M. D. & Mitchell, T. J. (1995) Exploratory designs for computer experiments, *Journal of Statistical Planning and Inference*, 43, pp. 381–402.

Myers, R. H. & Montgomery, D. C. (1995) *Response Surface Methodology: Process and Product Optimization using Designed Experiments* (Wiley).

Park, J. S. (1994) Optimal latin hypercube designs for computer experiments, *Journal of Statistical Planning and Inference*, 39, pp. 95–111.

Sacks, J., Schiller, S. B. & Welch, W. J. (1989) Designs for computer experiments, *Technometrics*, 34, pp. 15–25.

Shewry, M. & Wynn, H. (1987) Maximum entropy design, *Journal of Applied Statistics*, 14, pp. 165–170.

Tang, B. (1993) Orthogonal-array-based latin hypercubes, *Journal of the Americal Statistical Association*, 88, pp. 1392–1397.

Tang, B. X. (1994) A theorem for selecting oa-based latin hypercubes using a distance criterion, *Communications in Statistics—Theory and Methods*, 23, pp. 2047–2058.

Ye, K. Q. (1998) Column orthogonal latin hypercubes and their application in computer experiments, *Journal of the Americal Statistical Association*, 93, pp. 1430–1439.

Ye, K. Q., Li, W. & Sudjianto, A. (2000) Algorithmic construction of optimal symmetric latin hypercube designs, *Journal of Statistical Planning and Inference*, 90, pp. 145–159.

Zhu, C., Byrd, R. H., Lu, P. & Nocedal, J. (1994) L-BFGS-B: a limited memory FORTRAN code for solving bound constrained optimization problems. Technical Report, NAM-11, EECS Department, Northwestern University.

Appendix

Our simulated annealing algorithm for searching orthogonal-array-based latin hypercube designs is given below. Here, t_0 refers to the initial temperature and t the current temperature whilst FAC_t represents the rate at which temperature

reduction occurs and $t_{\min}$ the minimum temperature in the schedule. $I_{\max}$ represents the number of perturbations the algorithm will try without improvement before the temperature is reduced. Finally, ϕ represents the criterion we use to assess different latin hypercube designs. In the results presented here,

$$\sum_{i=1}^{n} \sum_{j=i+1}^{m} \frac{1}{d_{ij}^2}$$

is taken as in equation (1), however other choices are clearly possible.

Step 1. *Initialization*
Define t_0, $I_{\max}$, FAC_t, $t_{\min}$ and the orthogonal array. Randomly select an initial orthogonal-array latin hypercube design D using equation (2). Set $D_{\text{best}} = D$, $t = t_0$.

Step 2. *Temperature loop*
Set $FLAG = 0$, $I = 1$.

Step 3. *Perturbation loop*
Set D_{try} to D. Randomly select a column of D_{try}, then randomly select any element within this column. Randomly choose a second element in this column whose orthogonal array entry agrees with the first. Exchange these elements.

Step 4. If $\phi(D_{\text{try}}) < \phi(D)$, or with probability $e^{-(\phi(D) - \phi(D_{\text{try}}))/t}$, set D to D_{try} and $FLAG$ to 1.

Step 5. If $\phi(D_{\text{try}}) < \phi(D_{\text{best}})$ set D_{best} to D_{try} and I to 1. Otherwise increment I by 1.

Step 6. If $I < I_{\max}$ branch to Step 3.

Step 7. If $FLAG = 1$ and $t > t_{\min}$ multiply t by FAC_t and branch to Step 2.

Step 8. Stop and report D_{best}.