

**SELC : Sequential Elimination of Level Combinations
by Means of Modified Genetic Algorithms**

ABHYUDAY MANDAL

C.F. JEFF WU

School of Industrial and Systems Engineering

Georgia Institute of Technology

Atlanta, GA 30332-0205

(*amandal@isye.gatech.edu*)

(*jeffwu@isye.gatech.edu*)

KJELL JOHNSON

Pfizer Global Research and Development

Michigan Laboratories

Ann Arbor, MI 48105

(*Kjell.Johnson@pfizer.com*)

Abstract: To search for an optimum in a large search space, Wu, Mao, Ma (1990) suggested the SEL-method to find an optimal setting. Genetic algorithms (GAs) can be used to improve upon this method. To make the search procedure more efficient, new ideas of forbidden array and weighted mutation are introduced. Relaxing the condition of orthogonality, GAs are able to accommodate a variety of design points which allows more flexibility and enhances the chance of getting the best setting in fewer runs, particularly in the presence of interactions. The search procedure is enriched by a Bayesian method for identifying the important main effects and two-factor interactions. Illustration is given with the optimization of three functions, one of which is from Shekel's family.

Key words and phrases: Orthogonal arrays, fractional factorial designs, response surface methodology, Bayesian variable selection.

1. INTRODUCTION

In many scientific problems, the goal is to select an optimal candidate from a large pool of potential candidates. Genetic Algorithms (GAs) are a popular optimization technique when searching for global optimums. A modification of GAs, called SELC, is

proposed in this paper, which outperforms classical GAs in several practical situations. Here we present two scenarios where the SELC method can be useful. The first example is in the context of computer experiments, the second example arises in pharmaceutical industries.

In the last fifteen years many phenomena that could only be studied using physical experiments can now be studied by computer experiments. In a computer experiment, a deterministic output, $y(\mathbf{x})$, is computed for each set of input variables, \mathbf{x} , using numerical methods that are implemented by (complex) computer codes (Santner et al., 2003). In such cases, the complex function can be thought of as a “black box” and the proposed SELC method can be used to obtain the optimal settings efficiently. In Section 5, we illustrate how the SELC method can be efficiently used for a “black box” type problem.

The SELC method also has potential applications in the pharmaceutical industry. Within the past thirty years, technologies have been developed to explore and synthesize vast numbers of chemical entities. This technology, known as combinatorial chemistry, has been widely applied in the pharmaceutical industry, and is gaining interest in other areas of chemical manufacturing (Leach and Gillet, 2003, Gasteiger and Engel, 2003). In general, combinatorial chemistry identifies molecules that can be easily joined together, and employs robotics to physically make each molecular combination. Depending on the initial number of molecules, the number of combinations can be extremely large. For example, consider a core molecule onto which various reagents can be theoretically added to three locations. If one hundred reagents can be added at each location on the core, then one million potential products can be synthesized. In the pharmaceutical industry, combinatorial chemistry has been used to enhance the diversity of compound libraries, to explore specific regions of chemical space (i.e. focused library design), and to optimize one or more pharmaceutical endpoints such as activity, or ADMET (absorption, distribution, metabolism, excretion, toxicology) properties (Rouhi, 2003). While it is theoretically possible to make a large number of chemical combinations, it is generally not possible to follow-up on each newly synthesized entity. Instead of synthesizing all possible molecular combinations, combinatorial libraries are computationally created and evaluated using structure-based models. (For this purpose, specialized software uses “black box” type functions.) Then, a subset of promising compounds is selected for synthesis. For the purpose of optimization of pharmaceutical endpoints, the SELC method can be employed to efficiently find optimal molecules.

These real-life scenarios can be thought of as large dimensional design of experiment problems where the challenge is to identify the optimal design settings. Statistical design and analysis of experiments is an effective and commonly used tool in scientific and engineering investigation to understand and/or improve a system. Identifying important factors and choosing factor levels are among the first and most fundamental issues facing an experimenter. But, when confronted with a large number of important factors, designing an experiment can be difficult. Classical experimental design relies heavily on algebraic properties such as orthogonality. However, orthogonality does not allow the flexibility to accommodate all kinds of promising follow-up runs, which, in turn, makes finding suitable designs for large-scale problems difficult, particularly when the factors have more than two levels.

The use of high-fidelity computer simulations of physical phenomena (Bates et al., 1996) has stimulated new research into ways in which experimental design can be applied to such problems. One technique, motivated by design of experiments, was introduced by Wu, Mao, and Ma (1990) (hereafter abbreviated as WMM) known as Sequential Elimination of Levels (SEL). The idea of SEL is opposite to that of the “greedy algorithm” : instead of focusing on factor levels that improve the response, SEL focuses on those levels that worsen the response. Based on this idea, SEL eliminates one level of each factor in each sequence of the experiment. However, this kind of marginal analysis does not perform well in the presence of interactions, which is generally the case for high dimensional response surfaces. In this paper, the idea of SEL is extended to accommodate situations where important interactions are present. But, to make this accommodation, we must abandon follow-up designs that are orthogonal. Instead, a modified version of Genetic Algorithms (GAs) will be used to determine subsequent design points.

GAs have most often been viewed from a biological perspective. The metaphors of natural selection, cross breeding, and mutation have been helpful in providing a framework to explain how and why GAs work. Thus, it makes sense that most practical applications of GAs are rooted in the context of optimization. In an attempt to understand how GAs function as optimizers, Reeves and Wright (1999) considered GAs as a form of sequential experimental design. Recently, GAs have been used quite successfully in solving statistical problems, particularly for finding near optimal designs (Hamada et al. 2001, Heredia-Langner et al. 2003, 2004).

Our paper is organized as follows. In Section 2, we review the idea of SEL and classical GAs. A new version of SEL, called SELC is proposed in Section 3. The Bayesian model selection, which can be used in this process, is discussed in the Appendix. Behavior of the SELC algorithm is discussed in Section 4. In Section 5, our algorithm is applied to three functions, including one from Shekel's family, and the performance of this search methodology is investigated via simulations. Some concluding remarks are given in Section 6.

2. REVIEW : SEQUENTIAL ELIMINATION OF LEVELS AND GENETIC ALGORITHMS

SEL : WMM proposed their search method, based on orthogonal arrays, as follows :

1. For each factor, *eliminate* those level(s) with the worst mean value(s) of the performance measure computed from the current array.
2. Choose an orthogonal array (typically of a smaller size) for the *remaining* levels, and replace the array in step 1 with the new array.
3. Conduct another experiment on the *new* array.
4. Repeat steps 1 – 3 if necessary.

In step 1, if the mean is replaced by another descriptive statistic x (for example, minimum), the method is called SEL(x).

The main drawback of SEL is that its method of search is too restrictive for many optimization problems. First, for experiments that contain important interactions, the SEL method is not optimal because it eliminates individual levels of each factor. Hence, SEL can blindly eliminate a factor level that is required for the optimal run of the experiment. Second, SEL requires that subsequent experiments follow an orthogonal array. As mentioned previously, our modification of the SEL will prevent it from using an orthogonal array. In addition, orthogonal arrays are not flexible enough to handle complex response surfaces. To overcome this problem, we have developed a modified GA to determine subsequent design points.

GAs : Before describing the novel approach to improve SEL, we shall briefly review GAs (Holland, 1975). GAs are stochastic optimization tools that work on "Darwinian" models of population biology and are capable of obtaining near-optimal solutions for

multivariate functions without the usual mathematical requirements of strict continuity, differentiability, convexity or other properties. The algorithm attempts to mimic the natural evolution of a population by allowing solutions to reproduce, creating new solutions, and to compete for survival in the next iteration. The idea of GAs are to get “better solutions” using “good solutions”, and the main steps are given below :

1. *Solution representation* : For problems that require real number solutions, a simple binary representation is used where unique binary integers are mapped onto some range of the real line. Each bit is called a *gene* and this binary representation is called *chromosome*.

Once a representation is chosen, the GA proceeds as follows. A large initial population of random candidate solutions is generated in this representation; these are then continually transformed following steps 2 and 3.

2. *Select* the best and eliminate the worst solution on the basis of a fitness criterion (e.g., higher the better for a maximization problem) to generate the next population of candidate solutions.
3. *Reproduce* to transform the population into another set of solutions by applying the genetic operations “crossover” and “mutation”.
 - (a) **Crossover** : A pair of binary integers (chromosomes) are split at a random position and the head of one is combined with the tail of other and vice-versa.
 - (b) **Mutation** : The state (0 or 1) of a randomly chosen bit is changed. This helps the search not to get trapped into local optima.
4. *Repeat* steps (2) and (3) until some convergence criterion is met or some fixed number of generations has passed.

This algorithm has been shown to converge by Holland (1992), who first proposed this procedure in its most abstract form and discussed it in relation to adaptive and nonlinear systems.

3. SEQUENTIAL ELIMINATION OF LEVEL COMBINATIONS (SELC)

The main drawback of SEL is that its search is too restrictive. This method eliminates a level on the basis of marginal means which can be affected by the presence of

interactions. In order to overcome this drawback, we propose eliminating level combinations instead of just a single level. This modification is capable of capturing important interactions and provides more flexibility in the choice of follow-up design points. Our modification of SEL, Sequential Elimination of Level Combinations (SELC), incorporates the fundamentally new ideas of the forbidden array and weighted mutation. In Section 4, we shall see how these two novel concepts, motivated by the ideas of Design of Experiments, make the search algorithm much more efficient than classical GAs.

Recall that by the *effect hierarchy* principle (Wu and Hamada, 2000), two-factor interactions are more important than higher order interactions. In SELC, we employ this principle by allowing the algorithm to identify important interactions with respect to the optimization problem. Here we propose to eliminate those factor settings which have the same level combinations as that of the *worst* one for two factors. For larger dimensions, third or higher order tuples may need to be considered. The worst observed runs are stored in the *forbidden array* as the search procedure continues. New experiments are conducted with runs suggested by the SELC algorithm, which uses the idea of GAs, and promising level settings for a new run are achieved by using better runs from the previous experiments. Before formally defining the SELC algorithm, we define the concepts of the *forbidden array* and *weighted mutation*, both required by the algorithm. We end this section with a constructed example to illustrate the SELC algorithm.

Forbidden array: To avoid eliminating important interactions, we define the *forbidden array*. In the SELC method, instead of eliminating specific factor levels based on the results of an orthogonal array, we eliminate from future runs factor level *combinations* that are the same as those of the worst run(s) among current design points. At each stage of the experiment, the worst run(s) are chosen with probability governed by a “fitness” measure (i.e., value of y) and are stored in the *forbidden array*. Furthermore, we specify the *strength* and *order* of the forbidden array. The number of runs placed into the forbidden array at each sequence of the experiment defines the array’s *strength*. More specifically, a forbidden array of strength s contains the level combinations of the s worst runs of the experiment at each stage of the iterations. In addition, the runs stored in the forbidden array define a set of level combinations that will be prohibited from subsequent runs of the experiment. The number of level combinations that are prohibited from subsequent experiments defines the *order* of the forbidden array. A forbidden array of order k implies that any combinations of k or more levels from any

array in the forbidden array will be prohibited from being used in subsequent runs of the experiment. Thus, as the order decreases, the number of forbidden design points increases. Consequently, the forbidden array is the generating set of all runs which are forbidden by SELC.

For example, consider an experiment in which the goal is to maximize a response. Suppose the experiment has four factors, each at three levels (0, 1, and 2) and we choose a forbidden array with strength 1 and order 2. Further, suppose that the minimum value of $E(y)$ occurs when all factors are set to 0, and this design point is run during the experiment. When this run is placed into the forbidden array, it will prevent any design points with *two* or more factors set to level 0 (order=2). Note that only one member will be added to the forbidden array at each step (strength = 1).

Here the special case of $k = 1$ corresponds to the SEL method of WMM. Also, $s = 1$ corresponds to SEL(mini) of WMM. However, unlike the SEL-approach, the choice of worst run is probabilistic in SELC. In Section 6, we will illustrate how the choice of strength affects the performance of the search procedure.

After constructing the forbidden array, SELC starts searching for better level settings. The search procedure is motivated by GAs. The first step, as discussed in the review of GAs, is *solution representation*. Here the runs are looked upon as chromosomes. For an m -level factor, the levels are denoted by $0, 1, \dots, m - 1$. For example, for a 3^4 experiment, the design points(chromosomes) would take the form $(0, 0, 0, 0)$, $(0, 0, 0, 1)$, \dots , $(2, 2, 2, 2)$. *Unlike classical GAs, the chromosomes are not required to be binary arrays*. Next we identify, with probability proportional to the “fitness”, i.e. the value of y , the best runs to produce offspring of the next generation. After the good candidates are identified, they *reproduce* to generate potentially better candidates. In SELC, crossover is performed in the usual way, as explained in Section 2, but a modification is proposed for mutation.

Weighted mutation: In a generic GAs, genes mutate with an equivalent specified probability. Hence, the mutation rate does not incorporate other information gathered from prior knowledge about the system. For the SELC, we propose the use of prior information for generating mutation probabilities. For instance, suppose we know that the factor, F , has a significant main effect and *no* significant two-factor interactions. Then, we will change the level of this factor to a new level, l , with probability p_l , where

$$p_l \propto \bar{y}(F = l). \quad (3.1)$$

Next, suppose that factors F_1 and F_2 have a significant interaction. Then, the mutation should have a joint probability on F_1 and F_2 . That is, the mutation will occur if either F_1 or F_2 is randomly selected. Factor F_1 will be set to level l_1 and factor F_2 to level l_2 with probability q_{l_1, l_2} , where

$$q_{l_1, l_2} \propto \bar{y}(F_1 = l_1, F_2 = l_2). \quad (3.2)$$

If the selected factor does not have significant main effects or interactions, then its value is changed to any admissible levels with equal probability. Note that if the aim is to minimize $E(y)$, then the probabilities in (3.1) and (3.2) should be inversely proportional to \bar{y} .

A linear regression model can be used to identify the significant effects. But, a better, more time consuming approach is to consider a Bayesian variable selection strategy which is discussed in the Appendix. This method is used in the analysis illustrated at the end of this section.

Starting Design : The starting design is an orthogonal array, which allows us to efficiently estimate factor effects used in the process of weighted mutation. However, as the search proceeds, unlike SEL, the orthogonal structure of the design matrix will not be retained. Nonorthogonality is justified because the follow-up designs should be more flexible than the starting one, utilizing the information already at hand.

The SELC algorithm:

Initialize the design. Find an appropriate orthogonal array.

1. Conduct the experiment. Stop when the stopping criterion is achieved. (See below).
2. Construct the *forbidden array* and choose its strength and order.
3. Generate new *offspring*.
 - *Select* offspring for reproduction with probability proportional to their “fitness.”
 - *Crossover* the offspring.
 - *Mutate* the positions using *weighted mutation*.
4. Check the *new offspring's eligibility*. An offspring is *eligible* if it is not prohibited by any of the members of the forbidden array. If the offspring is eligible, go to step 1. If the offspring is ineligible, then repeat step 3.

Stopping Rules : The stopping rule is subjective. As the runs are added, the experimenter can decide, in a sequential manner, whether significant progress has been made towards optimization. Sometimes a target value or near optimum is predetermined for the experiment. Once the target is attained, the search can be stopped. But, typically, the number of experiments is limited by the resources at hand. This is often the case for the combinatorial chemistry example discussed in the Introduction. Examples of Section 5 illustrates a situation in which an experiment is limited by number of runs.

To illustrate the SELC method, consider a hypothetical experiment with 9 factors (denoted by A-I) each at 3 levels. In this example (and throughout this paper), we use the *linear-quadratic system* for coding linear and quadratic effects (Wu and Hamada, 2000) in order to eliminate correlation among a factor's linear and quadratic components. The linear-quadratic coding is expressed as follows :

<i>level</i>	\longrightarrow	<i>linear</i>	<i>quadratic</i>
0		-1	1
1		0	-2
2		1	1

The response is generated from the following model :

$$y = 2 + (A + 2B - 3C + D + 2E - 2A^2 + 2B^2 + 1.5C^2 - 3AC + 2.5AE - BF - 2CG + DGI)^2 + \varepsilon,$$

where ε is the standard normal error. In this analysis, we only consider the linear and quadratic effects and linear-by-linear interactions. Our aim is to find a setting for which the expected value of y is maximized.

The starting design for the SELC is an orthogonal array, 9 columns of an $OA(243, 3^{20}, 3)$. In this example we use a forbidden array with $s=1$ and $k=6$, and use a weighted mutation with the Bayesian variable selection strategy. After choosing the first member of the forbidden array, the search for better level settings is continued via crossover and weighted mutation. Upon computing the posterior probabilities of C and BC , we find that these are much larger than the posterior probabilities of the other effects. According to the weighted mutation scheme, if factor B or C is randomly selected for mutation, we must evaluate $q_{l_1 l_2}$'s in (3.2). The $q_{l_1 l_2}$'s are given below :

Factors	$C = 1$	$C = 2$	$C = 3$
$B = 1$	0.0526	0.0556	0.1212
$B = 2$	0.0973	0.0524	0.0865
$B = 3$	0.2933	0.1368	0.1043

After getting the new offspring, we check for eligibility and the search continues. In this example, the search was stopped after 400 runs. The maximum value of y was 679.68, which corresponds to the level setting of the third best design point. Note that we have only evaluated 2.03% of all possible combinations.

4. A JUSTIFICATION OF CROSSOVER AND WEIGHTED MUTATION

Steps of crossover and weighted mutation may be better understood through the following analysis. Consider the problem of maximizing $K(\mathbf{x})$, $\mathbf{x} = (x_1, \dots, x_p)$, over $a_i \leq x_i \leq b_i$. Instead of solving the p -dimensional maximization problem

$$\max \left\{ K(\mathbf{x}) : a_i \leq x_i \leq b_i, i = 1, \dots, p \right\}, \quad (4.1)$$

the following p one-dimensional maximization problems are considered,

$$\max \left\{ K_i(x_i) : a_i \leq x_i \leq b_i, i = 1, \dots, p \right\}, \quad (4.2)$$

where $K_i(x_i)$ is the i th marginal function of $K(\mathbf{x})$,

$$K_i(x_i) = \int K(\mathbf{x}) \prod_{j \neq i} dx_j \quad (4.3)$$

and the integral is taken over the intervals $[a_j, b_j]$, $j \neq i$. If the x_i in (4.1) and (4.2) can take only a finite number of values (discrete x_i), the integral in (4.3) is replaced by a finite sum. Let x_i^* be a solution to the i th problem in (4.2). The combination $\mathbf{x}^* = (x_1^*, \dots, x_p^*)$ may be proposed as an approximate solution to (4.1). A sufficient condition for \mathbf{x}^* to be a solution of (4.1) is that $K(\mathbf{x})$ can be represented as

$$K(\mathbf{x}) = \psi \left(K_1(x_1), \dots, K_p(x_p) \right) \quad (4.4)$$

and

ψ is nondecreasing in each K_i .

A special case of (4.4), which is of particular interest to statisticians, is

$$K(\mathbf{x}) = \sum_{i=1}^p \alpha_i K_i(x_i) + \sum_{i=1}^p \sum_{j=1}^p \lambda_{ij} K_i(x_i) K_j(x_j). \quad (4.5)$$

If λ_{ij} is nonzero, then SEL will have difficulty finding the optimal solution. However, SELC is more flexible and is better suited to find the optimal solution.

While the SEL method emphasizes on orthogonal arrays, SELC does not. The basic nature of GAs does not allow us to retain the orthogonal structure of the design. Though orthogonal arrays are good for estimating the factorial effects, they are not available for every combination of factor levels and for every run size. GAs do not require orthogonality and hence are more flexible in exploring new design points. This flexibility enhances the chance of getting the best setting in relatively fewer runs. If the response surface is very smooth, then any standard design and analysis should find the optimal settings. However, for many problems the response surface is not smooth. For instance, if the surface is undulated with local maxima and minima, the SELC method can perform well. The random nature of GA-type search explores the whole surface rapidly, while the weighted mutation uses prior knowledge about the surface to wisely direct the search.

The convergence of classical GAs was provided by Holland (1975) using the concept of schema. The SELC method makes a significant amount of modification to classical GAs and it is not obvious that the modifications proposed meet the requirements for convergence in Holland's paper. However, the simulation studies provided in the next section are quite convincing about the convergence.

5. EXAMPLES

We investigate the performance of SELC via several diverse simulations. Three different "ill-behaved" functions are considered and the effects of the fine tunings are illustrated through a variety of examples. For all these examples, we have the following settings. For crossover, after choosing one position randomly, parent chromosomes are split at that position and the left fragment of the first parent chromosome is combined with the right fragment of the other one to produce the first offspring and similarly for the other offspring. After that, a mutation-location is chosen randomly and weighted mutation is performed as described in Section 3. For comparison, some simulations have been done with usual mutation. This is referred to as "Unweighted Mutation". In

those cases, the level of the factor selected is changed randomly to any other admissible levels. For all simulations, the population size is taken to be 20.

Example 1 : Shekel 4 function (SQRIN)

The function

$$y(x_1, \dots, x_4) = \sum_{i=1}^m \frac{1}{\sum_{j=1}^4 (x_j - a_{ij})^2 + c_i}$$

is known as Shekel's function (Dixon and Szego, 1978), where the quantities $\{a_{ij}\}$ and $\{c_i\}$ are given in Table 1. The region of interest is $0 \leq x_j \leq 10$ and only integer values are considered. This function is one of the "black box" functions of computer experiments, discussed in the Introduction.

Table 1 : Coefficients for Shekel's function ($m = 7$)

i	$a_{ij}, j = 1, \dots, 4$				c_i
1	4.0	4.0	4.0	4.0	0.1
2	1.0	1.0	1.0	1.0	0.2
3	8.0	8.0	8.0	8.0	0.2
4	6.0	6.0	6.0	6.0	0.4
5	3.0	7.0	3.0	7.0	0.4
6	2.0	9.0	2.0	9.0	0.6
7	5.0	5.0	3.0	3.0	0.3

This set-up corresponds to an experiment with four factors each at 11 levels (i.e. the 11 integers). The starting design is an orthogonal array of 242 runs which is obtained by choosing 4 columns from the $OA(242, 11^{23})$ (Hedayat et al., 1999). In this example, unlike Section 3, Bayesian variable selection strategy was not used. In each step, Gibbs sampling consumes significant amount of time which would make it extremely difficult to run thousands of simulations. Instead, regression analysis is used to identify the important factors (at 5% level of significance). Forbidden arrays of order 3 are considered because order 1 or 2 becomes too restrictive for this problem by forbidding too many runs (and also, the results are not satisfactory). The results are compared with those of a random search and with simple GA.

Table 2 summarizes the results. "Random Search" corresponds to a design where all runs are selected randomly. "Random Followup" stands for a design, where the search begins with the same starting design and follow-up runs are selected randomly. "Genetic Algo" stands for a classical GA where the runs are looked upon as chromosomes and crossovers and mutations are done in the usual way. Recall that GA corresponds

to a special case of SELC with strength 0 and unweighted mutation. “SELC (No Forbiddance)” refers to weighted mutation only, because in this case, forbidden array is set to be empty (i.e., strength = 0). On the other hand, “SELC (Unweighted Mutation)” refers to forbiddance only. Here unweighted mutation is performed instead of weighted mutation. Finally, “SELC (Weighted Mutation)” refers to the SELC method proposed in Section 3.

The performance of the search algorithm is measured by its ability to find the global maximum. We also include its performance on finding second through fifth best values, because these five values stand apart from the others on the response surface.

In the first simulation, the search is stopped after 1000 runs, which is 6.83% of all possible 11^4 runs (Figure 1). As seen from Figure 1, GA performs better than random searches and SELC performs better than GA. The values for “SELC (No Forbiddance)” show the beneficial effect of weighted mutation (here strength of the forbidden array is 0) and the values for “SELC (Unweighted Mutation)” show the beneficial effect of forbidden array. “SELC (No Forbiddance)” finds the maximum in 53% of the cases, as opposed to 48% of GA. On the other hand, “SELC (Unweighted Mutation)” has success rate 55.5%. Finally, when the power of both forbidden array and weighted mutation are explored, SELC performs satisfactorily in 57.8% of the times. The most benefits are achieved by considering the weighted mutation. This effect is even more pronounced in the next example.

As the strength of the forbidden array increases, the power of the search algorithm also increases. However, the strength cannot be increased arbitrarily, because it will then prohibit too many design points from being considered. It should also be noted that the improvement of the performance of SELC with the increment of the strength is not so prominent for the same function when smaller run sizes are considered. In the second case, the search is stopped after 700 runs, and the improvements are not as significant. For Shekel 4 function, evolutionary algorithms would take larger runs to reap the benefits.

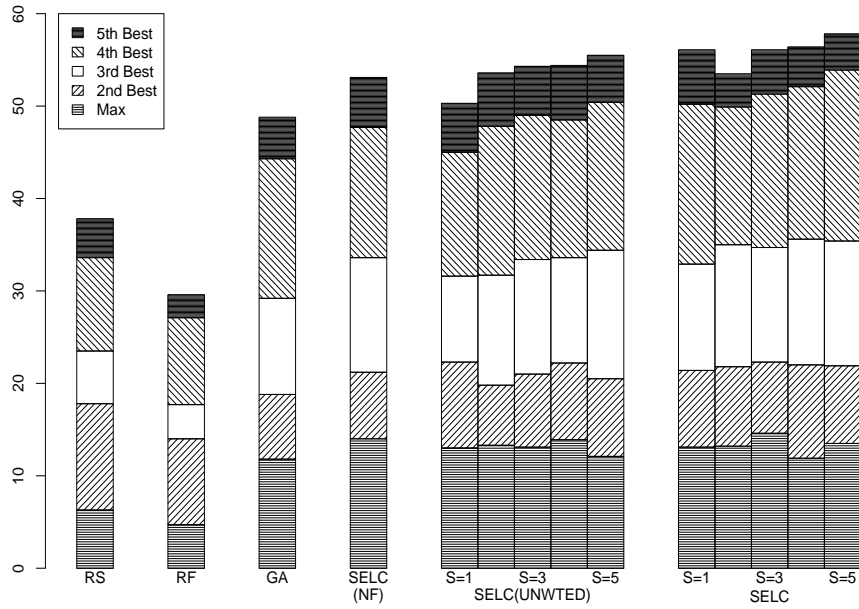


Figure 1 : Shekel Function : % of success in identifying global maximum for different methods (run size = 1000) [RS = Random Search, RF = Random Followup, GA = Genetic Algo, SELC(NF) = SELC(No Forbiddance), SELC(UNWTED) = SELC(Unweighted Mutation), SELC = SELC(Weighted Mutation), S = Strength]

Table 2 : % of success in identifying global maximum for different methods based on 1000 simulations (run size = 1000 and 700).

Strength		Max	2nd best	3rd best	4th best	5th best	Total
1000 RUNS							
Random Search		6.3	11.5	5.7	10.1	4.2	37.8
Random Followup		4.7	9.3	3.7	9.4	2.5	29.6
Genetic Algo		11.8	7.0	10.4	15.1	4.5	48.4
SELC (No Forbiddance)		14.0	7.2	12.4	14.1	5.4	53.1
SELC (Unweighted Mutation)	1	13.0	9.3	9.3	13.4	5.3	50.3
	2	13.3	6.5	11.7	16.1	5.8	53.4
	3	13.1	7.9	12.4	15.6	5.3	54.3
	4	13.9	8.3	11.4	14.9	5.9	54.4
	5	12.1	8.4	13.9	16.0	5.1	55.5
SELC (Weighted Mutation)	1	13.1	8.3	11.5	17.3	5.9	56.1
	2	13.2	8.6	13.2	14.9	3.6	53.5
	3	14.6	7.7	12.4	16.6	4.8	56.1
	4	11.9	10.1	13.6	16.5	4.3	56.4
	5	13.5	8.4	13.5	18.5	3.9	57.8
700 RUNS							
Random Search		4.2	9.0	4.0	9.2	4.1	30.5
Random Followup		3.0	6.8	3.0	5.1	2.4	20.3
Genetic Algo		5.8	5.6	6.0	9.2	3.3	29.9
SELC (No Forbiddance)		5.4	4.7	7.2	11.3	4.8	33.4
SELC (Unweighted Mutation)	1	5.8	6.1	6.0	9.9	4.9	32.7
	2	6.4	4.3	4.6	10.1	5.7	31.1
	3	7.1	4.3	5.9	8.7	5.2	31.2
	4	7.6	4.1	6.0	11.5	4.7	33.9
	5	5.2	4.6	6.6	10.2	4.9	31.5
SELC (Weighted Mutation)	1	6.3	5.5	6.9	11.5	4.0	34.2
	2	6.6	4.9	7.2	10.6	3.1	32.4
	3	7.2	4.6	9.6	10.6	4.1	36.1
	4	5.9	5.9	7.0	10.7	3.3	32.8
	5	5.9	4.7	8.5	10.3	4.1	33.5

Example 2

Consider the function

$$y(x_1, \dots, x_4) = 1 + \{\beta'x + (\gamma'x)^2 + \eta'x \times \tau'x\}^2,$$

where the parameters are given in Table 3. The region of interest is $0 \leq x_j \leq 10$ and only integer values are considered. This choice is motivated by discussions in Section 4, especially (4.5).

Table 3 : Coefficients for the function in Example 2

β	γ	η	τ
1	-3	2	-5
-2	-4	-10	0
2	5	2	-5
-1	-6	4	0

As in Example 1, this set-up also corresponds to an experiment with four factors each at 11 levels. The simulations are done with two starting designs: orthogonal array of size 121 and 242, which are obtained by choosing four columns from $OA(121, 11^{12})$ and $OA(242, 11^{23})$ respectively (Hedayat et al., 1999). The results are summarized in Table 4 and also in Figure 2. The simulations are done for a total of 300, 500 and 1000 runs.

GA performs much better than random search. This example shows that forbiddance need not always enhance the performance. In fact, without weighted mutation, forbiddance alone (i.e., “SELC (Unweighted Mutation)”) can perform worse than GA. This means that good runs are located in the “neighborhood” of bad runs and the response surface $y(x_1, \dots, x_4)$ is very undulated. However, weighted mutation significantly improves the performance of SELC. The main advantage of using SELC is that it uses prior information to direct the GA, thus finding a near optimum more quickly. This effect is clearly demonstrated for smaller runs, namely, with total run size 300 and 500. If the search is continued long enough, this gap will be narrowed and SELC may not perform much better than GA. Consider the first case where the starting design is an orthogonal array of size 121. For 300 runs, GA finds the maximum in 15% of the cases whereas SELC (Weighted Mutation) finds it in more than 40% of the cases. For 500 runs, the values are 40% and 75%, respectively. Finally for 1000 runs, the success rates

are 80% and 97%, respectively. The ratio of the success rate decreases as the run size increases, which is not surprising because these kind of evolutionary algorithms eventually find the near optimal solution, if they are run long enough. However, SELC finds the optimal quickly.

For the second case, the starting design is an orthogonal array of size 242. Here, for a total run size 300, the evolutionary-type algorithms are not expected to perform well because only 58 follow-up runs are available. Even with these few follow-up runs, SELC (Weighted Mutation) finds the maximum in more than 15% of the cases. With larger run sizes, the performance of both GA and SELC improves, with SELC performing significantly better than GA.

The overall pattern of the performance of SELC for both starting designs are similar. Also for 1000 runs, the effect of starting design diminishes and the success rates are very close for both cases. Note that, for this example, starting with a 121-run design, with only 300 evaluations (2.05% of all possible 11^4 runs), SELC finds the global maximum in more than 40% of the cases.

Table 4 : % of success in identifying global maximum for different methods based on 1000 simulations

Total Run Size	Strength	121-Run Design			242-Run Design		
		300	500	1000	300	500	1000
Random Search		1.7	3.6	7.0	1.7	3.6	7.0
Random Followup		1.1	2.5	5.7	0.4	2.4	4.6
Genetic Algo		15.1	39.5	79.7	3.4	28.9	79.5
SELC (No Forbiddance)		43.7	76.7	97.8	16.7	68.3	97.5
SELC (Unweighted Mutation)	1	13.9	39.9	80.9	3.2	30.9	77.2
	2	13.2	38.9	83.4	3.6	30.1	79.6
	3	17.0	41.4	82.3	4.3	31.4	78.4
	4	15.4	40.2	81.1	3.7	28.3	76.9
	5	15.0	44.1	81.5	3.6	29.5	78.5
SELC (Weighted Mutation)	1	41.3	76.9	97.3	17.1	67.4	98.4
	2	42.2	76.5	97.3	15.5	65.4	96.8
	3	40.5	75.1	98.2	15.7	67.6	97.8
	4	40.5	75.9	98.0	15.7	69.6	98.0
	5	39.9	73.9	97.9	18.2	66.1	96.9

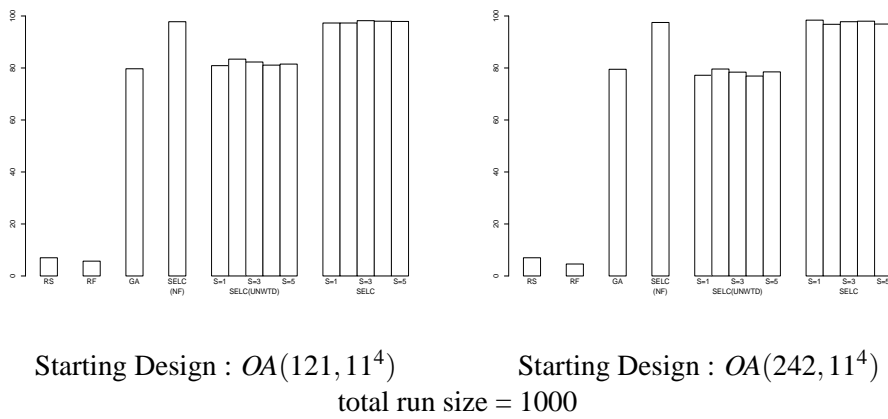
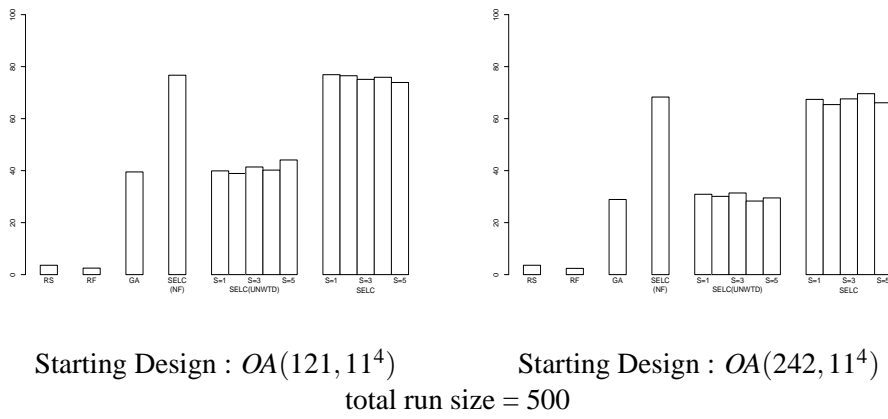
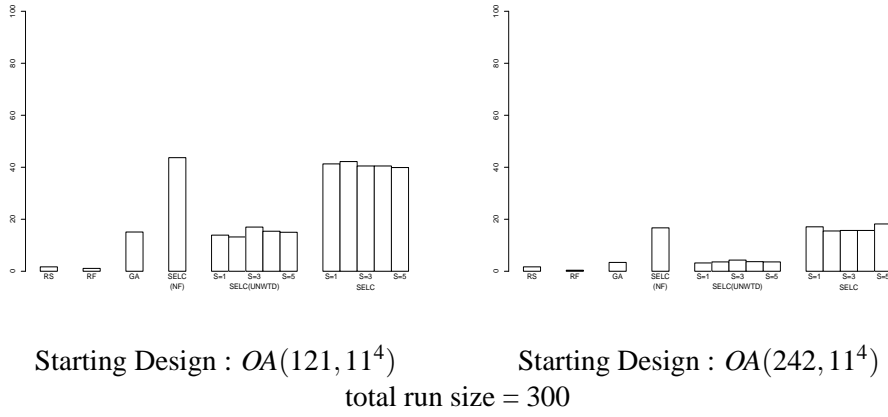


Figure 2 : % of success in identifying global maximum for different methods

Example 3

Levy and Montalvo (1985) provide the following function:

$$y(x_1, \dots, x_n) = \sin^2 \left\{ \pi \left(\frac{x_i + 2}{4} \right) \right\} + \sum_{i=1}^{n-1} \left(\frac{x_i - 2}{4} \right)^2 \left\{ 1 + 10 \sin^2 \left(\pi \left(\frac{x_i + 2}{4} \right) + 1 \right) \right\} \\ + \left(\frac{x_n - 2}{4} \right)^2 \left\{ 1 + \sin^2 (2\pi(x_n - 1)) \right\}.$$

Here $n = 4$ and only integer values of x_i 's ($0 \leq x_i \leq 10$) are considered. This again corresponds to an experiment with four factors each at 11 levels. Results are summarized in Table 5. Here the performance of SELC is quite similar to that of Example 2. Note that the analytic nature of the test function is quite different from that of the previous two examples. It is a standard test function in global optimization literature and is presented here to demonstrate the satisfactory performance of the SELC method over a variety of test functions.

Table 5 : % of success in identifying global maximum for different methods based on 1000 simulations

Strength		121-Run Design			242-Run Design		
		300	500	1000	300	500	1000
Total Run Size		300	500	1000	300	500	1000
Random Search		5.8	9.3	18.4	5.0	9.3	18.4
Random Followup		2.9	7.7	15.5	2.9	7.7	15.5
Genetic Algo		16.8	43.1	80.7	2.9	33.3	81.8
SELC (No Forbiddance)		30.3	62.2	94.5	5.9	50.6	93.8
SELC (Unweighted Mutation)	1	17.6	43.1	84.5	2.9	31.6	82.2
	2	16.7	42.9	84.3	3.3	32.4	82.0
	3	18.5	44.5	83.5	4.7	33.6	83.4
	4	21.2	44.1	83.9	3.4	33.4	81.9
	5	16.6	47.5	83.9	3.8	34.0	84.5
SELC (Weighted Mutation)	1	28.4	66.2	94.4	6.6	45.9	93.5
	2	26.0	66.2	92.8	7.5	50.5	91.8
	3	31.1	63.5	92.2	7.2	49.6	93.7
	4	29.4	63.8	90.1	7.6	46.8	91.2
	5	31.9	65.3	86.1	7.1	46.9	91.3

Examples 1 and 3 are from standard test functions in the global optimization literature. By closely examining those functions, one may have some idea about the location of the global maximum and may be able to save some computations. However, in many real life examples, (e.g., computer experiments) the analytic form of the function is either unknown or very complicated. In these situations, the function can be thought of as a “black box” and SELC method should perform well.

6. SUMMARY AND CONCLUSIONS

The problem of searching for an optimal design setting in a relatively large space is not easy. The SELC method does this job efficiently. Relaxing the condition of orthogonality, GA is flexible enough to explore more design points, which enhances the chance of finding the best setting in relatively fewer runs, particularly in the presence of interaction effects.

The by-product of SELC algorithm, discussed in the Appendix is also of interest. If there are many factors, the experimenter can get an insight by employing the Bayesian approach. The posterior probabilities identify the important factors and interactions clearly. This approach will result in a more comprehensive search of the model space. A system can have a large number of factors, of which only a handful are important. A major use of experimental design is screening, in which experimenters seek to identify significant effects (both main effects and potentially interactions) from a large set of candidate effects. The Bayesian variable selection helps in identifying the important factors and understanding the impact of a large number of factors in relatively fewer runs.

The novel idea of forbidden array and weighted mutation enables SELC to find the optimal solution more efficiently than GA. The improvement on performance, however, depends on the nature of the response surface. If the response surface is very smooth, any reasonable search algorithm should work satisfactorily. For an extremely complicated surface, almost complete enumeration might be needed irrespective of the efficiency of the search methods. For response surfaces whose ruggedness lies in between the two, SELC is expected to perform well.

ACKNOWLEDGEMENTS

The first author is thankful to Derek Bingham of Simon Fraser University for his

valuable suggestions and comments. Support to the first author at the Statistics Department of University of Michigan and Pfizer Global Research and Development, Ann Arbor Laboratories is gratefully acknowledged. The research is supported by NSF grant DMS-0305996.

REFERENCES

- Bates, R. A., Buke, R. J., Riccomagno, E., and Wynn, H. P. (1996), "Experimental design and observation for large systems", *J. R. Statist. Soc. B*, 58, 77-94.
- Chipman, H. (1996), "Bayesian Variable Selection with Related Predictors", *Canadian Journal of Statistics*, 24, 17-36.
- Chipman, H., Hamada, M., and Wu, C. F. J. (1997) "A Bayesian Variable Selection Approach for Analyzing Designed Experiments with Complex Aliasing", *Technometrics*, 39, 372-381.
- Dixon, L. C. W., and Szego, G. P. (1978). "The global optimization problem: an introduction," In *Towards Global Optimization 2* (Edited by L. C. W. Dixon and G. P. Szego), 1-15. North Holland, Amsterdam.
- Gasteiger, J. and Engel, T. (eds.) (2003) *Chemoinformatics : a Textbook*, Wiley-VCH, Weinheim.
- George, E. I., and McCulloch, R. E. (1993), "Variable selection Via Gibbs Sampling", *Journal of the American Statistical Association*, 88, 881-889.
- Hamada, M., Martz, H.D., Reese, C.S. and Wilson, A.G. (2001). "Finding Near Optimal Bayesian Designs Via Genetic Algorithms," *The American Statistician* 55, 175-181.
- Hedayat, A. S., Sloane, N. J. A., and Stufken, J. (1999), *Orthogonal Arrays : Theory and Applications*, Springer-Verlag, New York, Inc.
- Heredia-Langner, A., Carlyle, W.M., Montgomery, D.C., Borror, C.M. and Runger, G.C. (2003). "Genetic algorithms for the construction of D-optimal designs." *Journal of Quality Technology* 35, 28-46.

- Heredia-Langner, A., Montgomery, D. C., Carlyle, W. M., Borrer, C. M. (2004), "Model-Robust Optimal Designs: A Genetic Algorithm Approach", *Journal of Quality Technology*, 36, 3, 263–279.
- Holland, J. M. (1975), *Adaptation in natural and artificial systems*, The University of Michigan Press, Ann Arbor.
- Holland, J. M. (1992), *Adaptation in natural and artificial systems*, The MIT Press, Cambridge, MA.
- Leach, A. R., and Gillet, V. J. (2003) *An Introduction to Chemoinformatics*, Kluwer Academic Publishers, London.
- Levy, A. V., and Montalvo, A. (1985), "The tunnelling algorithm for the global minimization of functions", *SIAM Journal of Scientific and Statistical Computing*, 6, 15–29.
- Reeves, C. L., and Wright, C. C. (1999), "Genetic algorithms and the design of experiments", In Davis, L. D.; DeJong, K.; Vose, M. D. and Whitley, L. D. (1999) (Ed.), Springer-Verlag, New York, Inc. pp 207-226.
- Rouhi, A. M. (2003), "Custom Synthesis for Drug Discovery", *Chemical & Engineering News*, 81 , 7, 75–78.
- Santner, T. J., Williams B. J., and Notz, W. (2003) *The Design and Analysis of Computer Experiments*, Springer-Verlag, New York, Inc.
- Wu, C. F. J., Mao, S. S. and Ma, F. S. (1990), "SEL : A search method based on orthogonal arrays", In S. Ghosh (1990), (Ed.), *Statistical Design and Analysis of Industrial Experiments*, Marcel Dekker, Inc., New York, 279 - 310.
- Wu, C. F. J., and Hamada, M. (2000) *Experiments: Planning, Analysis, and Parameter Design Optimization*, John Wiley & Sons.

APPENDIX

Identification of Significant factors : A Bayesian Approach

The model selection problem amounts to identifying a subset of predictors as active, and in this setting there are typically more parameters to estimate than unique treatments.

Here we propose stochastic variable selection, based on the Gibbs sampler. We start with the given design and the corresponding responses. For the linear regression with normal errors,

$$y = X\beta + \sigma\varepsilon, \quad \varepsilon \sim N(0, 1), \quad (\text{A.1})$$

where β contains linear and quadratic main effects and linear-by-linear interaction effects. The Bayesian framework of Chipman, Hamada and Wu (1997) approaches model selection as follows. Importance of effects is captured via an unobserved vector δ of zeros and ones where $\delta_i = I\{\theta_i \neq 0\}$. A normal mixture prior is used for the coefficients β :

$$f(\beta_i|\delta_i) = \begin{cases} N(0, \tau_i^2) & \text{if } \delta_i = 0, \\ N(0, (c_i\tau_i)^2) & \text{if } \delta_i = 1. \end{cases} \quad (\text{A.2})$$

When $\delta_i = 0$, β_i has a high mass around zero and thereby, is not likely to have a large effect. On the other hand, when $\delta_i = 1$ a large value of c_i ensures that the variable is likely to have a large influence.

Not all models are equally likely. Based on the principles of *effect sparsity*, *effect hierarchy* and *effect inheritance* (Wu and Hamada, 2000), we can distinguish between the “likely” and “unlikely” models. Note that the commonly used independence prior, which implies that the importance of one factor is independent of that of another, is not very attractive as there are quadratic main and linear-by-linear interaction effects. Instead, we have used hierarchical priors, motivated by Chipman (1996). Consider a simple example with three main effects, A, B and C, each having three levels. It is logical to think that the importance of the interaction effect AB will depend on the importance of main factors A and B only. Also, the quadratic effect of level A will less likely be important if the linear effect of A is not important. This belief can be expressed in the prior for $\delta = (\delta_A, \delta_B, \delta_C, \delta_{A^2}, \delta_{B^2}, \delta_{C^2}, \delta_{AB}, \delta_{AC}, \delta_{BC})$ as follows :

$$\begin{aligned}
P(\delta) &= P(\delta_A, \delta_B, \delta_C, \delta_{A^2}, \delta_{B^2}, \delta_{C^2}, \delta_{AB}, \delta_{AC}, \delta_{BC}) \\
&= P(\delta_A, \delta_B, \delta_C)P(\delta_{A^2}, \delta_{B^2}, \delta_{C^2}|\delta_A, \delta_B, \delta_C)P(\delta_{AB}, \delta_{AC}, \delta_{BC}|\delta_A, \delta_B, \delta_C) \\
&= P(\delta_A)P(\delta_B)P(\delta_C)P(\delta_{A^2}|\delta_A, \delta_B, \delta_C)P(\delta_{B^2}|\delta_A, \delta_B, \delta_C)P(\delta_{C^2}|\delta_A, \delta_B, \delta_C) \\
&\quad P(\delta_{AB}|\delta_A, \delta_B, \delta_C)P(\delta_{AC}|\delta_A, \delta_B, \delta_C)P(\delta_{BC}|\delta_A, \delta_B, \delta_C) \\
&= P(\delta_A)P(\delta_B)P(\delta_C)P(\delta_{A^2}|\delta_A)P(\delta_{B^2}|\delta_B)P(\delta_{C^2}|\delta_C) \\
&\quad P(\delta_{AB}|\delta_A, \delta_B)P(\delta_{AC}|\delta_A, \delta_C)P(\delta_{BC}|\delta_B, \delta_C).
\end{aligned}$$

The first equality comes from the *conditional independence principle* which assumes that the higher order terms are independent when conditioned on the first order terms. Also it is assumed that first order terms are independent. The *inheritance principle* assumes that the importance of a higher order term depends only on its lower order parents. The nature of the exact dependence, which is followed in all our analysis, is given next:

$$P(\delta_A = 1) = p, \quad (\text{A.3})$$

$$P(\delta_{A^2} = 1|\delta_A) = \begin{cases} 0.1p & \text{if } \delta_A = 0, \\ p & \text{if } \delta_A = 1, \end{cases} \quad (\text{A.4})$$

$$P(\delta_{AB} = 1|\delta_A, \delta_B) = \begin{cases} 0.1p & \text{if } \delta_A + \delta_B = 0, \\ 0.5p & \text{if } \delta_A + \delta_B = 1, \\ p & \text{if } \delta_A + \delta_B = 2. \end{cases} \quad (\text{A.5})$$

In our analysis, we choose $p = 0.25$. A prior must also be specified for σ . Following George and McCulloch (1993), we take

$$\sigma^2 \sim IG(\nu/2, \nu\lambda/2)$$

where IG denotes the inverted gamma distribution. It can be shown that, $\nu\lambda/\sigma^2 \sim \chi_\nu^2$.

In addition, following George and McCulloch (1993), we take

$$\tau_j = \frac{\Delta y}{3\Delta X_j},$$

where Δy represents a “small” change in y , and ΔX_j represents a large change in X_j . In our examples, $\Delta X_j = \max(X_j) - \min(X_j)$ and $\Delta y = \sqrt{\text{Var}(y)}/5$. For priors of σ , $\nu = 5$ and $\lambda = \text{Var}(y)/25$ is used.

The posterior probabilities of β 's are computed using a Gibbs sampler.