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

Abhyuday Mandal **C. F. Jeff Wu** **Kjell Johnson**

Abstract

To search for an optimal design in a large search space, Wu, Mao, Ma (1990) suggested the SEL-method. Genetic algorithms (GA) can be used to improve upon this method. New ideas of forbidden array and weighted mutation are introduced. Relaxing the condition of orthogonality, GA is able to accommodate a variety of design points and enhances the chance of getting the best setting in fewer runs, particularly in the presence of interactions.

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

1. INTRODUCTION

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 such technique was introduced by Wu, Mao, and Ma (1990) (hereafter abbreviated as WMM) known as Sequential Elimination of Levels (SEL). The idea is, in some sense, opposite to that of the "greedy algorithm". Instead of focusing on factor levels that improve the response, SEL focuses on those that worsen the response. Based on this idea, SEL eliminates one level for 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 Algorithm (GA) 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 they 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 an optimizer, Reeves and Wright (1999) considered GAs as a form of sequential experimental design.

This paper is organized as follows. In Section 2, we review the idea of SEL and classical GA. 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 statistics 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.

GA : Before describing the novel approach to improve SEL, we shall briefly review Genetic Algorithms (GA) (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 GA is to get "better solutions" using "good Solutions". The main steps of GA 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".

   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.

   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, we propose elimination of level combinations instead of just a single level. It is capable of capturing important interactions and provides more flexibility in the choice of follow-up design points. This method is called the sequential elimination of level combinations (SELC).

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 bad 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 GA, 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 introduce 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 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. The lower the order, the higher the forbiddance. Consequently, forbidden array is the generating set of all runs which are forbidden by SELC.

For example, consider an experiment 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. If this run is placed into the forbidden array, then 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 shall see how the choice of strength affects the performance of the search procedure.

Reproduction : After constructing the forbidden array, the search for better level settings is started. The search procedure is motivated by GA. The runs are looked upon as chromosomes. Unlike classical GA, the chromosomes are not required to be binary arrays. We pick up, with probability proportional to the ``fitness'', i.e. the value of y, the best runs to produce offspring of the next generation. Crossover is done in the usual way.

Weighted mutation: In a generic GA, genes mutate with an equivalent specified probability. Hence, the mutation rate does not incorporate other information gathered from prior information 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

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

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_{l1l2}, where

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

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 the probabilities in (3.1) and (3.2) should be inversely proportional to the $\bar{y}$ value if the aim is to minimize *E(y)*. A linear regression model can be used to identify the significant effects.

Starting Design : The starting design is an orthogonal array. It allows us to efficiently estimate the factor effects which are 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. This 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. Begin with an orthogonal array, which allows the efficient estimation of factor effects used in the process of weighted mutation.

1. Conduct the experiment. Stop when the stopping criterion is achieved.

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 in the usual manner.
- *Mutate* the positions using *weighted mutation*.

4. Check the *new offspring's eligibility*. An offspring is called *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 one by one, the experimenter can decide, in a sequential manner, whether significant progress has been made and can stop after near optimum solution is attained. Sometimes a target value or near optimum is predetermined for the experiment. Once that target is attained, the search can be stopped. But, typically, the number of experiments is limited by the resources at hand.

5. EXAMPLES

The performance of SELC is investigated via simulations. Two different "ill-behaved" functions are considered and the effects of the fine tunings are illustrated through a variety of examples.

Example 1 : Shekel 4 function (SQRIN)

The function

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

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 $0 <= x_j <= 10$ and only integer values are considered.

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

i	$a_{ij}, j = 1,\ldots,4$				c_i
1	4	4	4	4	0.1
2	1	1	1	1	0.2
3	8	8	8	8	0.2
4	6	6	6	6	0.4
5	3	7	3	7	0.4
6	2	9	2	9	0.6
7	5	5	3	3	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 $OA(242,11^{23})$ (Hedayat et al., 1999). In this example, forbidden arrays of order 3 are considered as order 1 or 2 becomes too restrictive for this problem (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. Recall that GA corresponds to a special case of SELC with strength 0 and unweighted mutation. "SELC (Nof)", where "Nof" stands for "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 others on the response surface.

The search is stopped after 1000 runs, which is 6.83% of all possible 11^4 runs. As seen from Table 2, 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 (Unweigthed Mutation)" show the beneficial effect of forbidden array. "SELC (Nof)" finds the maximum in 53% of the cases, as opposed to 48% of GA. On the other hand, "SELC (Unweigthed 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 evident 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.

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

Strength		Max	2nd best	3rd best	4th best	5th best	Total
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
GA		11.8	7.0	10.4	15.1	4.5	48.4
SELC(Nof)		14.0	7.2	12.4	14.1	5.4	53.1
SELC (Unwtd Muta-tion)	1	13.0	9.3	9.3	13.4	5.3	50.3
	2	13.3	6.5	11.9	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 (Wtd Muta-tion)	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

Example 2

Consider the function

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

where the parameters are given in Table 3. The region of interest is 0<= x_j <= 10 and only integer values are considered.

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 earlier, 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. 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 $y(x_1,...,x_4)$ 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, these two numbers 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.

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. In spite of this, 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

Strength	121-Run Design			242-Run Design		
Total Run Size	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

GA		15.1	39.5	79.7	3.4	28.9	79.5
SELC(Nof)		43.7	76.7	97.8	16.7	68.3	97.5
SELC (Unwtd Muta-tion)	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 (Wtd Muta-tion)	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

Example 3

Levy and Montalvo (1985) provide the following function:

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

Here n=4 and only integer values of x_i's (0<= x_i <= 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.

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

Strength		121-Run Design			242-Run Design		
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
GA		16.8	43.1	80.7	2.9	33.3	81.8
SELC(Nof)		30.3	62.2	94.5	5.9	50.6	93.8
SELC (Unwtd Muta-tion)	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 (Wtd Muta-tion)	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., in computer experiment) 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 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 wiggled 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.

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.

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.

Hedayat, A. S., Sloane, N. J. A., and Stufken, J. (1999),*Orthogonal Arrays : Theory and Applications*, Springer-Verlag, New York, Inc.

Holland, J. M. (1975), *Adaptation in natural andartificial systems*, The University of Michigan Press, Ann Arbor.

Holland, J. M. (1992), *Adaptation in natural and artificial system*}, The MIT Press, Cambridge, MA.

Levy, A. V., and Montalvo, A. (1985), "The tunneling 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.

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.