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

Abhyuday MANDAL

Department of Statistics
University of Georgia
Athens, GA 30602-1952
(*amandal@stat.uga.edu*)

C. F. Jeff WU

School of Industrial and Systems Engineering
Georgia Institute of Technology
Atlanta, GA 30332-0205
(*jeffwu@isye.gatech.edu*)

Kjell JOHNSON

Pfizer Global Research and Development
Michigan Laboratories
Ann Arbor, MI 48105
(*Kjell.Johnson@pfizer.com*)

To search for an optimum in a large search space, Wu, Mao, and Ma suggested the sequential elimination of levels (SELs)-method to find an optimal setting. Genetic algorithms (GAs) can be used to improve on 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 likelihood 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. A real example on compound optimization is also given.

KEY WORDS: Bayesian variable selection; Fractional factorial designs; Orthogonal arrays; Response surface methodology.

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 in a large candidate or design space. A modification of GAs, called sequential elimination of level combinations (SELC), is proposed in this article and outperforms classical GAs in several practical situations. Here we present two scenarios in which the SELC method can be useful. The first scenario is in the context of computer experiments, and the second arises in pharmaceutical industries.

In the last 15 years, many phenomena that could be studied only 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 implemented by (complex) computer codes (Santner, Williams, and Notz 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. Over the past 30 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 uses 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 100 reagents can be added at each location on the core, then 1 million products potentially 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 target efficacy or ADMET (absorption, distribution, metabolism, excretion, toxicology) properties (Rouhi 2003). Although it is theoretically possible to make a large number of chemical combinations, it generally is not possible to follow up on each newly synthesized entity. Instead of synthesizing all possible molecular combinations, combinatorial libraries are created computationally and evaluated using structure-based models. (For this purpose, specialized software uses "black box"–type functions.) In addition, chemists look for reagent combinations known to produce undesirable compounds and attempt to avoid these combinations during synthesis. Using these constraints, a subset of promising reagents is

TECHNOMETRICS, MAY 2006, VOL. 48, NO. 2
DOI 10.1198/004017005000000526

selected to generate a combinatorial library. By construction, the SELC is a natural fit for searching for optimal molecules in combinatorial chemistry.

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 aimed at understanding and/or improving 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. But 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, Buke, Riccomagno, and Wynn 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 WMM) and called *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. But 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 article we extend the idea of SEL to accommodate situations in which important interactions are present. But to make this accommodation, we must abandon follow-up designs that are orthogonal, and instead use a modified version of genetic algorithms (GAs) to determine subsequent design points.

GAs most often have 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, Martz, Reese, and Wilson 2001; Heredia-Langner, Carlyle, Montgomery, Borror, and Runger 2003; Heredia-Langner, Montgomery, Carlyle, and Borror 2004).

The article is organized as follows. In Section 2 we review the idea of SEL and classical GAs. We propose new version of SEL, called SELC, in Section 3. (The Bayesian model selection, which can be used in this process, is discussed in the App.) We discuss the behavior of the SELC algorithm in Section 4. In Section 5 we apply the proposed algorithm to three functions, including one from Shekel's family, and investigate the performance of this search methodology via simulations. In Section 6 we use SELC to identify potentially good compounds for synthesization in a pharmaceutical industry. We give some concluding remarks in Section 7.

2. REVIEW: SEQUENTIAL ELIMINATION OF LEVELS AND GENETIC ALGORITHMS

2.1 Sequential Elimination of Levels

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

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

In step 1, if the mean is replaced by another descriptive statistic z (e.g., minimum), then the method is called SEL(z).

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 without considering interactions. 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.

2.2 Genetic Algorithms

Before describing the novel approach to improve SEL, we 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. The idea behind GAs is to get "better solutions" using "good solutions"; the algorithm process is as follows:

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 a *chromosome*.

   Once a representation is chosen, the GA proceeds as follows. A large initial population of random candidate solutions is generated; 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., the 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 of "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 the other and vice versa.
   b. Mutation. The state (0 or 1) of a randomly chosen bit is changed. This helps avoid the search being 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 was 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

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. 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 describe 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 use this principle by allowing the algorithm to identify important interactions with respect to the optimization problem. Here we propose to eliminate those factor settings that 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 to narrow the search space. 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.

3.1 Forbidden Array

In some situations, prior knowledge is available about certain factor-level combinations that lead to undesirable results. Consider the introductory combinatorial chemistry example. In this setting, chemists can often identify runs (i.e., new molecules), based on their scientific knowledge and prior experience, which are not worth creating in the laboratory. These runs can be placed into the forbidden array before initializing the SELC algorithm.

In the absence of prior knowledge, the SELC is initialized with an orthogonal design. The data from this initial experiment are then used to suggest run(s) that are not optimal. These run(s) are then placed into the forbidden array.

In subsequent steps 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 that are forbidden by SELC.

For example, consider an experiment in which the goal is to maximize a response. Suppose that 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 that 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. In addition $s = 1$ corresponds to SEL(mini) of WMM. However, unlike in the SEL approach, in SELC the choice of worst run is probabilistic. In Section 6 we 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 viewed as chromosomes. For an m-level factor, the levels are denoted by $0, 1, \ldots, m-1$. For example, for a 3^4 experiment, the design points (chromosomes) would take the form $(0, 0, 0, 0)$, $(0, 0, 0, 1), \ldots, (2, 2, 2, 2)$. *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 for producing 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.

3.2 Weighted Mutation

In a generic GA, genes mutate with an equivalent specified probability. Hence the mutation rate does not incorporate other

information gathered from previous knowledge about the system. For the SELC, we propose using 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

$$p_l \propto \bar{y}(F = l). \tag{1}$$

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 will be set to level l_2 with probability $q_{l_1 l_2}$, where

$$q_{l_1 l_2} \propto \bar{y}(F_1 = l_1, F_2 = l_2). \tag{2}$$

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 (1) and (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.

3.3 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 in SEL, the orthogonal structure of the design matrix is not retained. Nonorthogonality is justified because the follow-up designs should be more flexible than the starting design, using the information already at hand.

3.4 The SELC Algorithm

Initialize the design with an appropriate orthogonal array:

1. Conduct the experiment.

   Stop when the stopping criterion is achieved (see later).

2. Construct the *forbidden array* and choose its strength and order.
3. Generate b new *offspring*:
   a. Select offspring for reproduction with probability proportional to their "fitness."
   b. Crossover the offspring.
   c. Mutate the positions using weighted mutation.
4. Check eligibility. An offspring is *eligible* if it is not prohibited by any of the members of the forbidden array. If an offspring is ineligible, then discard and generate another new offspring.
5. If $b = 1$ and more than one offspring were generated, then randomly select one offspring for the experiment.

Depending on the situation, the SELC method can be *fully* ($b = 1$) or *batch* ($b = b$) *sequential*. For fully sequential SELC, a new eligible offspring is generated in each iteration and the experiment is conducted. For batch sequential SELC, a new set of eligible offspring is generated in each iteration and the experiment is conducted. Depending on the application, either fully sequential or batch sequential may be more suitable. For example, in combinatorial chemistry, a batch sequential SELC is more appropriate.

Fully sequential SELC method is used in the illustrative example of this section as well as in Section 6. In contrast, a batch sequential SELC method is used in Section 5. In the later case, a fixed number of offspring are generated before running the experiments to evaluate their performance.

3.5 Stopping Rules

The stopping rule is subjective and depends on progression of the algorithm and experimental constraints. As the runs are added, the experimenter can decide, in a sequential manner, whether significant progress has been made toward optimization. Sometimes a target value or near-optimum value 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 Section 1. Examples in Section 5 illustrate a situation in which an experiment is limited by number of runs.

To illustrate the SELC method, consider a hypothetical experiment with nine factors (denoted by A–I) each at three levels. In this example (and throughout this article), 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:

Level		*Linear*	*Quadratic*
0		−1	1
1	⟶	0	−2
2		1	1

The response is generated from the following model:

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

where ϵ is the standard normal error. In this analysis we consider only 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, nine columns of an OA$(243, 3^{20}, 3)$. Without having a prior knowledge about the unfavorable runs, here we use a forbidden array with $s = 1$ and $k = 6$, and also 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. After 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, then we must evaluate the $q_{l_1 l_2}$'s in (2). The $q_{l_1 l_2}$'s are as follows:

Factors	$C=1$	$C=2$	$C=3$
$B=1$	.0526	.0556	.1212
$B=2$	.0973	.0524	.0865
$B=3$	.2933	.1368	.1043

After generating the new offspring, we check for eligibility and the search continues. In this example, using the fully sequential version of SELC, 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 the SELC algorithm found this near-optimum design point by evaluating only 2.03% of all possible combinations.

4. A JUSTIFICATION OF CROSSOVER AND WEIGHTED MUTATION

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

$$\max\{K(\mathbf{x}) : a_i \le x_i \le b_i, i = 1, \ldots, p\}, \tag{3}$$

the following p one-dimensional maximization problems are considered:

$$\max\{K_i(x_i) : a_i \le x_i \le b_i, i = 1, \ldots, p\}, \tag{4}$$

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

$$K_i(x_i) = \int K(\mathbf{x}) \prod_{j \ne i} dx_j, \tag{5}$$

and the integral is taken over the intervals $[a_j, b_j], j \ne i$. If the x_i in (3) and (4) can take only a finite number of values (discrete x_i), then the integral in (5) is replaced by a finite sum. Let x_i^* be a solution to the ith problem in (4). The combination $\mathbf{x}^* = (x_1^*, \ldots, x_p^*)$ may be proposed as an approximate solution to (3). A sufficient condition for $\mathbf{x}^*$ to be a solution of (3) is that $K(\mathbf{x})$ can be represented as

$$K(\mathbf{x}) = \psi\big(K_1(x_1), \ldots, K_p(x_p)\big) \tag{6}$$

and

$$\psi \text{ is nondecreasing in each } K_i.$$

A special case of (6) that is of particular interest to statisticians is

$$K(\mathbf{x}) = \sum_{i=1}^{p} \alpha_i K_i(x_i) + \sum_{i=1}^{p} \sum_{j=1}^{p} \lambda_{ij} K_i(x_i) K_j(x_j). \tag{7}$$

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

Whereas the SEL method emphasizes orthogonal arrays, SELC does not. The basic nature of GAs does not allow us to retain the orthogonal structure of the design. Although 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 for exploring new design points. This flexibility enhances the likelihood 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, then the SELC method can perform well. The random nature of the GA-type search explores the whole surface rapidly, whereas the weighted mutation uses previous 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 proposed modifications meet the requirements for convergence in Holland's work. 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. We consider three different "ill-behaved" functions, and illustrate the effects of the fine tunings through various examples. For all of these examples, we use 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 second parent chromosome to produce the first offspring. Then mutation locations are chosen randomly for each offspring, and weighted mutation is performed as described in Section 3. For comparison, some simulations have been done with unweighted mutation, which allows the level of the factor to be changed randomly to any other admissible level. For all simulations, the population size is 20, which corresponds to a batch size of 20 (i.e., $b = 20$). For each of these examples, we assume that there is no prior knowledge about undesirable runs. Hence we initialize the forbidden array using information gathered from the initial orthogonal array.

5.1 Example 1: Shekel 4 Function (SQRIN)

The function

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

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 $0 \le x_j \le 10$, and only integer values are considered. This function is one of the "black box" functions of computer experiments discussed in Section 1.

This setup corresponds to an experiment with 4 factors each at 11 levels (i.e., the 11 integers). The starting design is an orthogonal array of 242 runs obtained by choosing 4 columns from the OA($242, 11^{23}$) (Hedayat, Sloane, and Stufken 1999). In this example, unlike in Section 3, Bayesian variable selection

Table 1. Coefficients for Shekel's Function ($m = 7$)

i	$a_{ij}, j = 1, \ldots, 4$				c_i
1	4.0	4.0	4.0	4.0	.1
2	1.0	1.0	1.0	1.0	.2
3	8.0	8.0	8.0	8.0	.2
4	6.0	6.0	6.0	6.0	.4
5	3.0	7.0	3.0	7.0	.4
6	2.0	9.0	2.0	9.0	.6
7	5.0	5.0	3.0	3.0	.3

strategy was not used. In each step, Gibbs sampling consumes a significant amount of time, making 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 in which all runs are selected randomly. "Random follow-up" represents a design for which the search begins with the same starting design and follow-up runs are selected randomly. "Genetic algorithm" refers to a classical GA in which the runs are considered chromosomes and crossovers and mutations are performed in the traditional 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). In contrast, "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.

Table 2. Percent Success in Identifying Global Maximum for Different Methods Based on 1,000 Simulations (run size = 1,000 and 700)

	Strength	Maximum	Second best	Third best	Fourth best	Fifth best	Total
1,000 runs							
Random search		6.3	11.5	5.7	10.1	4.2	37.8
Random follow-up		4.7	9.3	3.7	9.4	2.5	29.6
Genetic algorithm		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 follow-up		3.0	6.8	3.0	5.1	2.4	20.3
Genetic algorithm		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

The performance of the search algorithm is measured by its ability to find the global maximum. We also include its performance on finding the 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 1,000 runs, which is 6.83% of all possible 11^4 runs (Fig. 1). As seen in Figure 1, GA performs better than random searches, and SELC performs better than GA. The values for SELC (no forbiddance) demonstrate the beneficial effect of weighted mutation (here the strength of the forbidden array is 0), and the values for SELC (unweighted mutation) demonstrate the beneficial effect of forbidden array. SELC (no forbiddance) finds the maximum in 53% of the cases, as opposed to 48% for GA. In contrast, SELC (unweighted mutation) has a success rate of 55.5%. Finally, when the power of both forbidden array and weighted mutation are simultaneously explored, SELC performs satisfactorily 57.8% of the time. The greatest benefits are achieved by considering the weighted mutation, and 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 SELC's performance 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 the Shekel 4 function, evolutionary algorithms would require more runs to reap the benefits.

5.2 Example 2

Consider the function

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

where the parameters are given in Table 3. The region of interest is $0 \le x_j \le 10$ and only integer values are considered. This choice is motivated by discussions in Section 4, especially by (7).

As in Example 1, this setup also corresponds to an experiment with 4 factors each at 11 levels. The simulations are done with two starting designs: orthogonal array of size 121 and 242, obtained by choosing four columns from $\mathrm{OA}(121, 11^{12})$ and $\mathrm{OA}(242, 11^{23})$ (Hedayat et al. 1999). The results are summarized in Table 4 and Figure 2. The simulations are done for a total of 300, 500, and 1,000 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 $y(x_1, \ldots, x_4)$ is very undulated. However, weighted mutation significantly

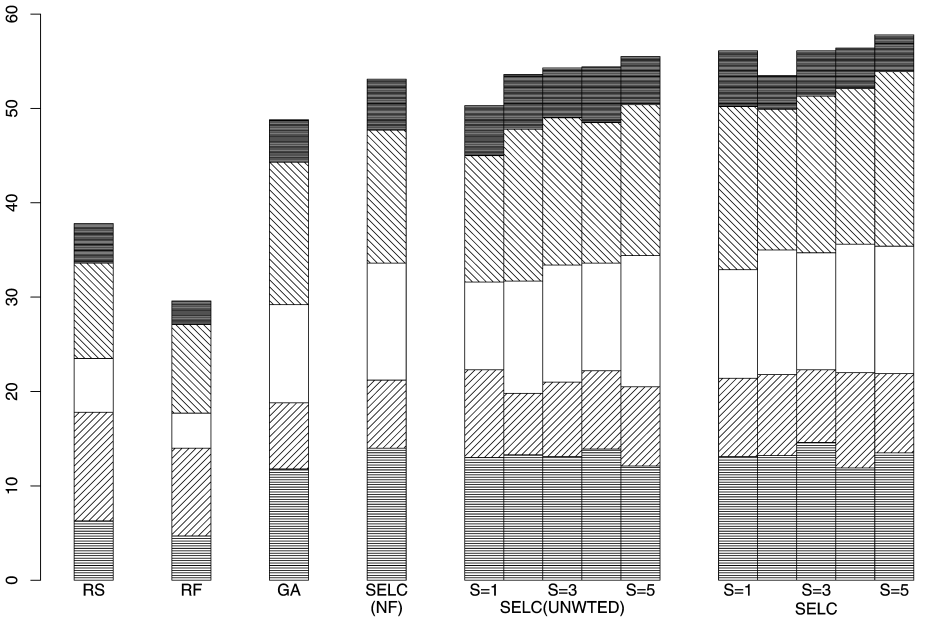

Figure 1. Shekel Function: Percent Success in Identifying Global Maximum for Different Methods. Run size = 1,000. RS = random search; RF = random follow-up; GA = genetic algorithm; SELC(NF) = SELC (no forbiddance); SELC(UNWTED) = SELC (unweighted mutation); SELC = SELC (weighted mutation); S = strength. (▬ 5th best; ▧ 4th best; □ 3rd best; ▨ 2nd best; ▤ maximum.)

improves the performance of SELC. The main advantage of SELC is that it uses prior information to direct the GA, thus finding a near-optimal solution more quickly. This effect is clearly demonstrated for smaller runs, those with total run sizes 300 and 500. If the search is continued long enough, then 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 cases, whereas SELC (weighted mutation) finds it in more than 40% of cases. For 500 runs, these values are 40% and 75%. Finally, for 1,000 runs, the success rates are 80% and 97%. 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. However, SELC finds the optimum 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 cases. With larger run sizes, the performance of both GA and SELC improves, with SELC performing significantly better than GA.

The overall patterns of performance of SELC for both starting designs are similar. Also for 1,000 runs, the effect of starting design diminishes, and the success rates are very close for both cases. Note that for the 121-run initial design, SELC finds the global maximum in more than 40% of the cases by evaluating only 300 evaluations (2.05% of all possible 11^4 runs).

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

5.3 Example 3

Levy and Montalvo (1985) provided the following function:

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

Table 4. Percent Success in Identifying Global Maximum for Different Methods Based on 1,000 Simulations

		121-run design			*242-run design*		
	Strength	*300*	*500*	*1,000*	*300*	*500*	*1,000*
Random search		1.7	3.6	7.0	1.7	3.6	7.0
Random follow-up		1.1	2.5	5.7	.4	2.4	4.6
Genetic algorithm		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

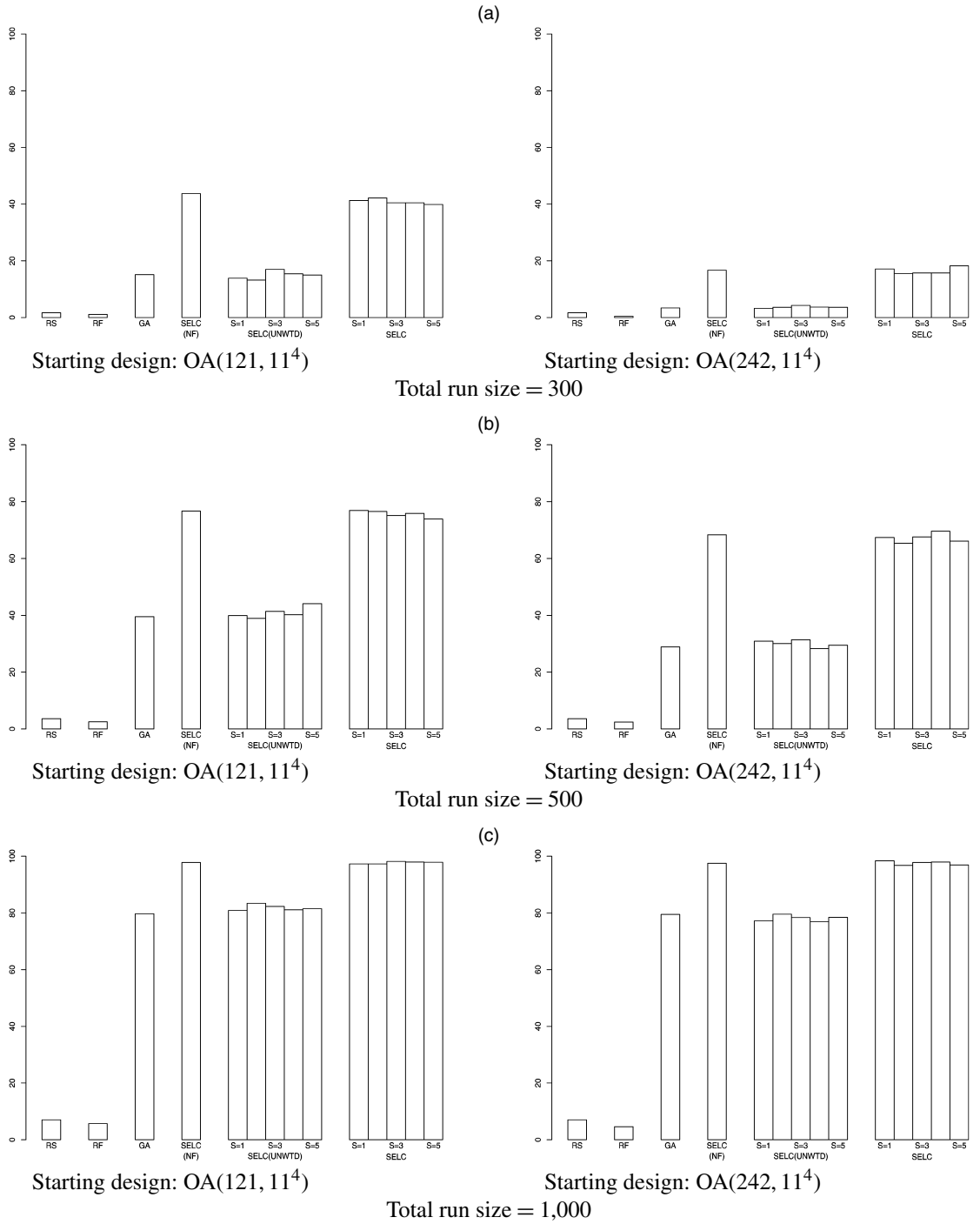

Figure 2. Percent Success in Identifying Global Maximum for Different Methods for Total Run Sizes (a) 300, (b) 500, and (c) 1,000.

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 4 factors each at 11 levels. The 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 Examples 1 and 2. It is a standard test function in global optimization literature and is presented here to demonstrate the satisfactory performance of the SELC method over various test functions.

Examples 1 and 3 are from standard test functions in the global optimization literature. By closely examining those functions, one may gain 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. APPLICATION

We applied the SELC method to a combinatorial chemistry problem in which a combination of reagents was desired to

Table 5. Percent Success in Identifying Global Maximum for Different Methods Based on 1,000 Simulations

		121-run design			242-run design		
	Strength	300	500	1,000	300	500	1,000
Random search		5.8	9.3	18.4	5.0	9.3	18.4
Random follow-up		2.9	7.7	15.5	2.9	7.7	15.5
Genetic algorithm		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

maximize target efficacy. In this example, target efficacy is measured by a compound's percent inhibition of activity for a specific biological screen. For this screen, a percent inhibition value of 50 or greater is an indicator of a promising compound, and percent inhibition values of 95 or greater have a high probability of exhibiting activity in confirmation screening.

Consider a core molecule onto which reagents can be added to three locations, denoted by A, B, and C. In this example, the desired compound space included two reagents at position A, 10 reagents at position B, and 14 reagents at position C. In total, the compound space contained 280 ($= 2 \times 10 \times 14$) possible chemical entities. The reagents in this application can be considered different levels of the factors (i.e., positions) and are denoted by integers, 1, 2, and so on. In this example, 208 of the 280 chemical entities were actually created without the assistance of the SELC algorithm. To demonstrate the algorithm's benefits to the combinatorial chemistry group, we applied the SELC to this problem under the hypothetical constraint that resources were limited to creating only 25 compounds.

Based on previous scientific knowledge, some combinations of reagents for this experiment were known to yield unfavorable percent inhibition values. We used these combinations of reagents to focus the initial starting design and placed them into the forbidden array before the experiment. Tables 6 and 7 present the relative frequency of occurrence of the individual levels of factors B and C in the forbidden array. Because we were limited to creating 25 total compounds, we chose a $2 \times 2 \times 3$ orthogonal array to initialize the experiment. Using Tables 6 and 7, in conjunction with scientific guidance, the initial orthogonal array included levels 8 and 9 of factor B and levels 3, 4, and 8 of factor C. The results from the initial orthogonal array are presented in Table 8 (upper half).

After completing the initial orthogonal array, we needed to choose subsequent design points. Because not all levels of factors B and C were explored in the initial experiment, the SELC algorithm was slightly modified to enable it to explore other

Table 6. Factor B

Level	1	2	3	4	5	6	7	8	9	10
Relative frequency (in %)	3	3	26	4	29	5	10	1	5	14

Table 7. Factor C

Level	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Relative frequency (in %)	8	7	7	4	5	4	4	3	8	5	16	11	8	8

parts of the chemistry space. Specifically, if a factor was found to be significantly associated with an improvement in response (at 5% level), the levels of that factor received probabilities proportional to their individual association with the response. However, if a factor was not significantly associated with improvement in response, then all levels of the factor received equal probability for the weighted mutation. In this application, factor B was significantly associated with the response after the 13th compound was created. The probabilities of mutation to the levels of factor B are

$$p_8 = \frac{24+34+63+2+5+49+83+56+14+83}{1{,}016} \times .75 + \frac{1}{10} \times .25,$$

$$p_9 = \frac{0+12+21+9+0+5}{1{,}016} \times .75 + \frac{1}{10} \times .25,$$

and

$$p_j = \frac{1}{10} \times .25 \quad \text{for all } j \neq 8, 9.$$

The denominator, 1,016, is the sum of positive responses, and the weights of .75 and .25 are arbitrary. The 10 levels of B account for the $1/10$ in the foregoing expression. As desired, to maximize the target efficacy, we considered only positive values of the response in calculating p_j's. The results from the subsequent runs of the experiment are given in Table 8 (bottom half). We used a fully sequential SELC method here.

We analyzed all compounds run in the experiment in a follow-up experiment in which their IC_{50} values were deter-

Table 8. Combinatorial Chemistry Example

No.	A	B	C	y	
1	1	8	8	24	
2	1	9	8	−23	
3	2	8	8	34	
4	2	9	8	12	
5	1	8	3	63	*
6	1	9	3	21	
7	2	8	3	2	
8	2	9	3	9	
9	1	8	4	5	
10	1	9	4	−16	
11	2	8	4	49	*
12	2	9	4	5	
13	2	8	10	83	*
14	2	3	4	65	*
15	2	1	4	107	*
16	2	2	10	49	
17	2	8	2	56	*
18	1	6	10	19	
19	2	2	4	60	*
20	2	10	10	39	
21	1	8	10	14	
22	2	6	8	90	*
23	2	6	10	64	*
24	2	1	1	−3	
25	2	2	5	63	*

mined. [IC_{50} assays (assays to determine the concentration of a drug-like compound resulting in a 50% reduction in activity of a disease target) are a commonly used method for assessing drug efficacy in pharmaceutical screening regimens. Typically, these assays are performed via serial dilutions of dimethyl sulfoxide (DMSO) compound libraries to achieve dilutions of 2×10^7 in 100% DMSO. Subsequent to the DMSO serial dilution, assays are performed with each dilution to ascertain the IC_{50} of the compound of interest.] Compounds that were judged to be acceptable by the chemists are indicated with an asterisk in Table 8. Clearly, the SELC method succeeded in identifying a rich set of promising compounds.

7. 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 likelihood of finding the best setting in relatively fewer runs, particularly in the presence of interaction effects. Because the forbidden array can make use of previous knowledge to rule out unfavorable settings, the SELC is particularly well suited for scientific problems in which such knowledge is available.

A byproduct of the SELC algorithm, discussed in the Appendix is also of interest. If there are many factors, the experimenter can get an insight by using the Bayesian approach. The posterior probabilities clearly identify the important factors and interactions. 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 depends on the nature of the response surface, however. If the response surface is very smooth, then 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.

ACKNOWLEDGMENTS

The authors thank Dirk Bornemeier of Pfizer Global Research and Development, Michigan Laboratories, who helped focus on the pharmaceutical application. The first author thanks Derek Bingham of Simon Fraser University for his valuable suggestions and comments. Thanks also go to the associate editor and referees for helpful comments. The research of the first two authors is supported by National Science Foundation grant DMS-03-05996. 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.

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,

$$y = X\beta + \sigma\epsilon, \qquad \epsilon \sim \mathrm{N}(0, 1), \tag{A.1}$$

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. The importance of effects is captured via an unobserved vector $\boldsymbol{\delta}$ of 0's and 1's where $\delta_i = I\{\theta_i \neq 0\}$. A normal mixture prior is used for the coefficients β,

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

When $\delta_i = 0$, β_i has a high mass around 0 and thus is not likely to have a large effect. In contrast, 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, because 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 only on the importance of main factors A and B, and also that the quadratic effect of level A will be less likely to be important if the linear effect of A is not important. This belief can be expressed in the prior for $\boldsymbol{\delta} = (\delta_A, \delta_B, \delta_C, \delta_{A^2}, \delta_{B^2}, \delta_{C^2}, \delta_{AB}, \delta_{AC}, \delta_{BC})$ as

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

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. In addition, 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

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

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

and

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

In our analysis, we choose $p = .25$. A prior must also be specified for σ. Following George and McCulloch (1993), we take

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

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

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 $\lambda = \text{var}(y)/25$ are used. The posterior probabilities of the β's are computed using a Gibbs sampler.

[Received March 2004. Revised May 2005.]

REFERENCES

Bates, R. A., Buke, R. J., Riccomagno, E., and Wynn, H. P. (1996), "Experimental Design and Observation for Large Systems," *Journal of the Royal Statistical Society*, Ser. 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*, eds. L. C. W. Dixon and G. P. Szego, Amsterdam: North Holland, pp. 1–15.

Gasteiger, J., and Engel, T. (eds.) (2003), *Chemoinformatics: A Textbook*, Weinheim: Wiley–VCH.

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*, New York: Springer-Verlag.

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., and Borror, C. M. (2004), "Model-Robust Optimal Designs: A Genetic Algorithm Approach," *Journal of Quality Technology*, 36, 263–279.

Holland, J. M. (1975), *Adaptation in Natural and Artificial Systems*, Ann Arbor, MI: University of Michigan Press.

——— (1992), *Adaptation in Natural and Artificial Systems*, Cambridge, MA: MIT Press.

Leach, A. R., and Gillet, V. J. (2003), *An Introduction to Chemoinformatics*, London: Kluwer Academic Publishers.

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 *Evolutionary Algorithms*, eds. L. D. Davis, K. DeJong, M. D. Vose, and L. D. Whitley, New York: Springer-Verlag, pp. 207–226.

Rouhi, A. M. (2003), "Custom Synthesis for Drug Discovery," *Chemical & Engineering News*, 81, 75–78.

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

Wu, C. F. J., and Hamada, M. (2000), *Experiments: Planning, Analysis, and Parameter Design Optimization*, New York: Wiley.

Wu, C. F. J., Mao, S. S., and Ma, F. S. (1990), "SEL: A Search Method Based on Orthogonal Arrays," in *Statistical Design and Analysis of Industrial Experiments*, ed. S. Ghosh, New York: Marcel Dekker, pp. 279–310.