A Genetic Algorithm Approach to Multiple-Response Optimization

FRANCISCO ORTIZ, JR., JAMES R. SIMPSON, AND JOSEPH J. PIGNATIELLO, JR.

Florida A&M University and Florida State University, Tallahassee, FL 32310

ALEJANDRO HEREDIA-LANGNER

Pacific Northwest National Laboratory, Richland, WA 99352

Many designed experiments require the simultaneous optimization of multiple responses. A common approach is to use a desirability function combined with an optimization algorithm to find the most desirable settings of the controllable factors. However, as the problem grows even moderately in either the number of factors or the number of responses, conventional optimization algorithms can fail to find the global optimum. An alternative approach is to use a heuristic search procedure such as a genetic algorithm (GA). This paper proposes and develops a multiple-response solution technique using a GA in conjunction with an unconstrained desirability function. The GA requires that several parameters be determined in order for the algorithm to operate effectively. We perform a robust designed experiment in order to tune the genetic algorithm to perform well regardless of the complexity of the multiple-response optimization problem. The performance of the proposed GA method is evaluated and compared with the performance of the method that combines the desirability with the generalized reduced gradient (GRG) optimization. The evaluation shows that only the proposed GA approach consistently and effectively solves multiple-response problems of varying complexity.

KEY WORDS: Heuristic Methods; Regression Modeling; Response Surface Desirability Functions.

Introduction and Background

THE EVALUATION of products or processes can involve the simultaneous study of several performance characteristics or responses. The task then becomes simultaneously optimizing each response of interest. This multiple-response optimization problem can be challenging, resulting in trade-offs associated with setting input parameters to achieve various response goals or targets. Ultimately, the goal is to determine the settings of the design variables such that the best possible combination of the responses is obtained.

There have been many creative methods discussed in the statistics literature for treating multiple-response problems. The performances of these techniques are dependent on or limited by the size and complexity of the problem. For problems where only a few factors and responses are involved, overlaying the contour plots of each respective response can help identify the regions where the goals are simultaneously met. An explanation of this graphical approach can be found in Myers and Montgomery (2002). This method has clear disadvantages, since it does not provide exact answers and, even for relatively small dimensional problem, can be quite difficult to analyze.

A more general approach that can be used is

Mr. Francisco Ortiz, Jr., is a doctoral student in the Department of Industrial Engineering. He is a member of ASQ. His email address is fortiz@eng.fsu.edu.

Dr. James R. Simpson is an Associate Professor in the Department of Industrial Engineering. He is a member of ASQ. His email address is simpson@eng.fsu.edu.

Dr. Joseph J. Pignatiello, Jr., is an Associate Professor in the Department of Industrial Engineering. He is a member of ASQ. His email address is pigna@eng.fsu.edu.

Dr. Alejandro Heredia-Langner is a postdoctoral fellow in the Statistical and Quantitative Sciences Group. His email address is Alejandro.Heredia-Langner@pnl.gov.

to formulate a multiple-response problem as a constrained optimization problem. This is done by selecting one of the responses as the objective function and treating the other responses as constraints. Choosing which one of the responses to be the objective function may be difficult in some cases, in particular when the number of responses is large. An example of this technique can be found in Del Castillo and Montgomery (1993).

A third technique used for solving multiple-response problems consists of using a method for combining multiple responses into a single value, followed by a numerical method to optimize the combined response function. The techniques available for combining multiple-response models into a single scalar include distance functions (Khuri and Conlon (1981)), squared error loss functions (Pignatiello (1993) and Vining (1998)), and desirability functions (Harrington (1965), Derringer and Suich (1980), and Del Castillo et al. (1996)). The desirability methods are easy to understand and implement, available in software, and provide flexibility in weighting individual responses.

The desirability approach consists of transforming m individual response functions each into desirabilities based on the particular goal for that response. Individual desirabilities $d_i(\hat{y}_i)$, $i = 1, 2, \ldots, m$ map response values to unit-less utilities bounded by $0 < d_i(\hat{y}_i) < 1$, where a higher $d_i(\hat{y}_i)$ value indicates that response value $\hat{y}_i$ is more desirable. Combining the individual desirability values usually involves using either a multiplicative or additive model and results in an overall desirability function associated with the vector of independent variables, $\mathbf{x}$. A common approach is to define the overall desirability as the geometric mean of individual desirabilities where

$$D(\mathbf{x}) = \left(d_1(\hat{y}_1)d_2(\hat{y}_2)\cdots d_m(\hat{y}_m)\right)^{1/m}. \qquad (1)$$

Although several forms have been proposed for $d_i(\hat{y}_i)$, the most commonly adopted are those of Derringer and Suich (1980). For a target value (two-sided) goal, the individual desirability is

$$d_i(\hat{y}_i) = \begin{cases} \left(\dfrac{\hat{y}_i - L_i}{T_i - L_i}\right)^{s_i}, & L_i \le \hat{y}_i \le T_i \\ \left(\dfrac{\hat{y}_i - H_i}{T_i - H_i}\right)^{t_i}, & T_i \le \hat{y}_i \le H_i \\ 0, & \text{otherwise} \end{cases} \qquad (2)$$

where L_i is the lower limit, H_i is the upper limit, and T_i is the target value for response i. The exponents t_i and s_i are weights that allow for linear ($s_i = t_i = 1$) or nonlinear behavior between a bound (L_i or H_i) and the target (T_i). For the maximization (one-sided) objective, $d_i(\hat{y}_i)$ becomes

$$d_i(\hat{y}_i) = \begin{cases} 0, & \hat{y}_i \le L_i \\ \left(\dfrac{\hat{y}_i - L_i}{H_i - L_i}\right)^{s_i}, & L_i \le \hat{y}_i \le H_i \\ 1, & \hat{y}_i \le H_i. \end{cases}$$

The response surfaces associated with these overall measures can be highly nonlinear and multimodal, requiring some optimization method capable of locating the global optimum.

If the response surface associated with $D(\mathbf{x})$ is fairly well behaved, conventional nonlinear optimization techniques can be used to find the most desirable $\mathbf{x}$. Current solution methods include search methods such as the Nelder–Mead simplex (Nelder and Mead (1965)) and gradient-based algorithms such as the generalized reduced gradient (GRG; see, e.g., Reklaitis et al. (1983)). However, as the number of responses that define quality and the number of significant factors increase, the corresponding $D(\mathbf{x})$ response surface can become highly nonlinear, multimodal, and heavily constrained. In these cases, conventional optimization algorithms and direct search methods may find only a local optimum or even fail to find a feasible solution (see Reklaitis et al. (1983)). Figure 1 displays a $D(\mathbf{x})$ that conventional optimization algorithms and direct search methods may find difficult to solve.

For these situations, one alternative is to use a heuristic search procedure (e.g., genetic algorithm, simulated annealing, or tabu search). The perfor-

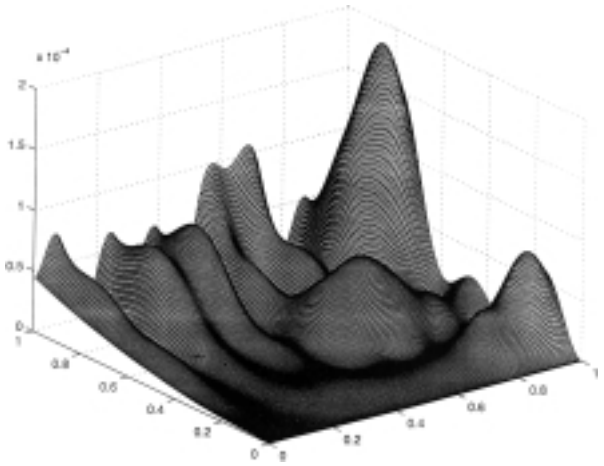


FIGURE 1. Example of a Multimodal Response Surface.

mances of these search procedures are problem specific. Some heuristics are easier to engineer to solve a particular problem. The genetic algorithm has been used to solve various statistical problems in quality engineering. Heredia-Langner et al. (2003) and Borkowski (2003) both used GAs for constructing designs, and Carlyle et al. (2000) mentions using genetic algorithms to solve complex multiple-response optimization problems.

A genetic algorithm is a search technique that is based on the principles of natural selection or survival of the fittest. Holland (1974) developed the ideas and concepts behind the GA and many authors have refined his initial approach. Instead of using gradient information, the GA uses the objective function directly in the search. The genetic algorithm searches the solution space by maintaining a population of potential solutions. Then, using evolving operations such as recombination, mutation, and selection, the GA creates successive generations of solutions that will evolve and take on the positive characteristics of their parents and thus gradually approach optimal or near-optimal solutions. By using the objective function directly in the search, genetic algorithms can be effectively applied in nonconvex, highly nonlinear, complex problems (Goldberg (1989)). The genetic algorithm is not guaranteed to find the global optimum, but it is less likely to get trapped at a local optimum than traditional gradient-based search methods when the objective function is not smooth and generally well behaved. Our findings also indicate that the genetic algorithm examines a larger portion of the solution space than conventional methods and is therefore more likely to find feasible solutions in heavily constrained problems. A brief description of common procedures and parameters used in the GA can be found in the appendix.

A drawback of the genetic algorithm is the large number of parameters that must be tuned to obtain optimal performance. The initial sample size, convergence criterion, and other implementation parameters all affect the performance of the genetic algorithm. However, a robust parameter design approach could be used to determine these parameter settings in the presence of varied multiple-response scenarios.

This paper focuses on the development of a method for performing multiple-response optimization in the presence of a moderate to large number of responses. The basic approach consists of an unconstrained desirability function combined with genetic algorithm optimization. This proposed genetic algorithm approach is then tuned for application in multiple-response optimization using robust parameter design. The goal of this paper is to develop a multiple-response optimization method that performs well across a variety of commonly encountered problem scenarios. We will demonstrate how a robust parameter design approach can be used to enhance a method that effectively optimizes multiple responses in several different problem environments. We will also discuss the relationships between genetic algorithm parameters and the characteristics that define a problem's complexity. The performance of the proposed method will be compared with an optimization method commonly used when dealing with multiple responses.

Developing the Multiple-Response Performance Metric

The initial step for implementing a genetic algorithm for multiple-response optimization is to establish an encoding structure for potential solutions. The results of designed experiments are typically expressed in terms of individual response regression models that relate factors to the output performance measures. Dimensionless coded factors, $x_1, \ldots, x_k$, are commonly used such that the response surface is estimated over the region defined by $-1 < x_i < 1$ for $i = 1, 2, \ldots, k$. For this study, a GA *chromosome* represents the vector of coded factors, $\mathbf{x} = [x_1, x_2, \ldots, x_k]$, where we will refer to each entry in this vector as a *gene*. An example of a chromosome for an 8-factor multiple-response process is $\mathbf{x} = [-0.451, 0.346, 0.518, 0.701, -0.573, -0.634, 0.937, -0.448]$.

Each chromosome is associated with a vector of m fitted responses, $\hat{\mathbf{y}} = [\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_m]'$, for which the genetic algorithm fitness is determined by an overall desirability function $D(\mathbf{x})$. Hence, the metric for evaluating the GA chromosomes, fitness, is just the combined response overall desirability value for a given $\mathbf{x}$.

An Unconstrained Desirability Function for a Genetic Algorithm

The multiplicative overall desirability function results in $D(\mathbf{x}) = 0$ if any $d_i(\hat{y}_i) = 0$, that is, if any $\hat{y}_i$ is either infeasible or otherwise undesirable. This strict penalization does not allow the genetic algorithm to maintain infeasible $\mathbf{x}$ (chromosomes), which may perform well in some subset of the $d_i(\hat{y}_i)$, even though some of the genes may be highly desirable. Thus, the use of the multiplicative overall desirability function

can limit the search capabilities of the GA. Additive overall desirability functions, $D(\mathbf{x})$, such as the arithmetic mean discussed in Kros and Mastrangelo (2001), incorporate a weak penalty when a constraint has been violated. However, this method for combining the $d_i(\hat{y}_i)$ can result in infeasible solutions that have higher desirability values than feasible solutions. Consequently, the corresponding optimization procedure may converge to infeasible solutions. To take full advantage of the search capabilities of the genetic algorithm, this paper proposes a new unconstrained multiplicative desirability function capable of discriminating among infeasible settings of $\mathbf{x}$, while not allowing infeasible solutions to have higher overall desirability values than feasible solutions.

This proposed unconstrained desirability function satisfies two criteria essential for the genetic algorithm to perform well. First, it allows the genetic algorithm to evaluate infeasible solutions in order to locate regions of feasibility. Second, it properly differentiates between feasible and infeasible solutions. To enable the genetic algorithm to evaluate infeasible $D(\mathbf{x})$, we propose incorporating a penalty function into the overall desirability function, a fairly common tool when using genetic algorithms. This method involves incorporating the constraints directly into the overall desirability (or fitness) function. The proposed overall desirability function, $D^*(\mathbf{x}) = D_{\mathrm{DS}}(\mathbf{x}) - P(\mathbf{x})$, incorporates penalties through $P(\mathbf{x})$, which is proportional to the square of the constraint violation. Here, $D_{\mathrm{DS}}(\mathbf{x})$ refers to the Derringer and Suich overall desirability, as in Equation (1), in which individual desirabilities are combined using the geometric mean.

The overall penalty function $P(\mathbf{x})$ is also a combined function of the individual fitted responses, reflecting the overall severity of the infeasibility. The overall penalty function is

$$P(\mathbf{x}) = \left[\left(p_1(\hat{y}_1) p_2(\hat{y}_2) \cdots p_m(\hat{y}_m) \right)^{1/m} - c \right]^2,$$

where the corresponding individual penalties $p_i(\hat{y}_i)$ are

$$p_i(\hat{y}_i) = \begin{cases} c + \left| \dfrac{\hat{y}_i - L_i}{T_i - L_i} \right|, & -\infty \le \hat{y}_i \le L_i \\ c, & L_i \le \hat{y}_i \le H_i \\ c - \left| \dfrac{\hat{y}_i - H_i}{T_i - H_i} \right|, & H_i \le \hat{y}_i \le +\infty \end{cases}$$

and c is a relatively small constant used to force $p_i(\hat{y}_i) > 0$. Requiring a nonzero $p_i(\hat{y}_i)$ ensures that some nonzero overall penalty $P(\mathbf{x})$ is assessed for each infeasible solution. For this study, $c = 0.0001$ is used. Smaller or larger values of c can be used without loss of generality. Incorporating this overall penalty function into a combined fitted response metric, the proposed overall desirability function becomes

$$D^*(\mathbf{x}) = [d_i(\mathbf{x}) \cdots d_m(\mathbf{x})]^{1/m} - \left\{ [p_i(\mathbf{x}) \cdots p_m(\mathbf{x})]^{1/m} - c \right\}^2. \quad (3)$$

The purpose accomplished by the overall penalty function is to ensure design space locations $\mathbf{x}$ are appropriately ranked relative to each other based on their degree of infeasibility. The genetic algorithm will be guided by this information so that feasible solutions can be quickly identified. Once feasible solutions are found, their associated penalty function becomes $P(\mathbf{x}) = 0$, removing any penalty function influence from $D^*(\mathbf{x})$.

Comparison of Desirability Functions within the Genetic Algorithm

To demonstrate the need to generate an overall desirability function capable of evaluating infeasible solutions, consider the following illustration. Figure 2 shows a comparison of the Derringer and Suich overall desirability function with the unconstrained overall desirability function in Equation (3) using genetic algorithm optimization on a multiple response problem.

Genetic algorithms iteratively generate subsets of candidate design space points for evaluation by the overall desirability function. Each subset of evaluated design space points is called a generation. In this example, the $D^*(\mathbf{x})$ function led to identifying a fea-

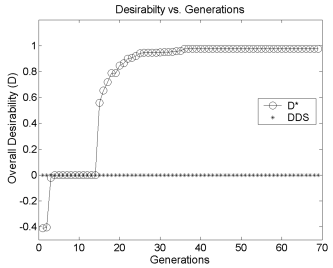


FIGURE 2. Comparison of GA Optimization Between the Derringer and Suich Desirability Function (D_{DS}) and the Unconstrained Desirability Function (D^*).

TABLE 1. Comparison Between $D_{\mathrm{DS}}(\mathbf{x})$ and $D^*(\mathbf{x})$ for Two Infeasible Design Points

Design Space Point (a)							
					$D_{\mathrm{DS}}(\mathbf{x})$	$D^*(\mathbf{x})$	
Responses	Value	Low	Target	High	d_i	d_i	p_i
$\hat{y}_1$	60	50	60	70	1	1	0.000001
$\hat{y}_2$	75	50	75	100	1	1	0.000001
$\hat{y}_3$	50	47	50	53	1	1	0.000001
$\hat{y}_4$	30	40	45	50	0	0	2.000001
			Overall Desirability		0		−1.34E-09

Design Space Point (b)							
					$D_{\mathrm{DS}}(\mathbf{x})$	$D^*(\mathbf{x})$	
Responses	Value	Low	Target	High	d_i	d_i	p_i (penalty)
$\hat{y}_1$	60	50	60	70	1	1	0.000001
$\hat{y}_2$	45	50	75	100	0	0	0.200001
$\hat{y}_3$	60	47	50	53	0	0	2.333334
$\hat{y}_4$	30	40	45	50	0	0	2.000001
			Overall Desirability		0		−9.66E-04

sible region in only 14 generations and final convergence after 35 generations. In contrast, using $D_{\mathrm{DS}}(\mathbf{x})$ as the measure of overall desirability in the GA, the algorithm was unable to find a feasible solution after 50 generations. In both cases, the GA initiated its search with a population of infeasible $\mathbf{x}$ vectors. Because the $D_{\mathrm{DS}}(\mathbf{x})$ approach fails to distinguish the degree of infeasibility, no useful information is passed on to the GA so that it can find a feasible $\mathbf{x}$.

To demonstrate how $D^*(\mathbf{x})$ can provide more information than $D_{\mathrm{DS}}(\mathbf{x})$ in terms of GA optimization, the overall desirability function formulations are applied to two design space points. Table 1 illustrates a situation where two different $\mathbf{x}$ vectors both produce infeasible solutions and $D_{\mathrm{DS}}(\mathbf{x}) = 0$. However, the $\mathbf{x}$ in (a) has three of the four fitted responses at their target value, whereas the $\mathbf{x}$ in (b) has three of the four fitted responses infeasible. The design space point in (a) is more likely to be closer to the feasible region and therefore is a better candidate for the GA in future generations. The proposed overall desirability function is able to differentiate between the two infeasible $\mathbf{x}$ vectors, awarding a higher $D^*(\mathbf{x})$ to the preferred $\mathbf{x}$ in (a).

Designing Genetic Algorithms for Multiple-Response Optimization

The GA is sensitive to the many choices of parameters. In this paper, a designed experiment is performed to determine the best GA parameters settings (control variables) under various response surface conditions (noise variables). The following sections detail the considerations associated with tuning the GA in the presence of a variety of multiple-response scenarios.

Evaluation of Evolutionary Operators: Tuning the Genetic Algorithm

The genetic algorithm uses an approach that commonly involves starting with a random selection of design space points. The genes of these initial indi-

viduals are combined in meaningful ways to produce new solutions, and these are evaluated and ranked by objective function value. At this point, solutions determined to have better fitness are selected to remain in the population and, among these, some experience alterations (recombination and mutations) that allow the algorithm to explore new regions of the problem space. Each phase in a GA run involves the use and the selection of numerous components. Although the GA is usually robust to the values chosen for these components, the task of finding those that will perform well in a variety of circumstances is not easy and can become a difficult optimization problem in itself. We will now discuss parameter setting alternatives that will be part of an experimental design to find the GA that performs best for various multiple-response optimization problems.

Parent Population

The initial set of design space locations evaluated at the start of each generation is called the parent population in GA literature. In principle, the size of the population affects both the quality of the solution obtained and the efficiency of the GA. If the population size is too small, not enough information about the entire search space will be obtained. Therefore, the GA may fail to find the neighborhood of the global optimum and thus may converge to a mediocre solution. A large population size allows the GA to perform a more thorough search. However, because a large population requires more objective function evaluations, the rate of convergence will be slower.

Real-valued encoding is a relatively new methodology within GAs. Most of the methods used to identify a potential range for population size deal with binary encoding (Reed et al. (2000)). For real-valued encoding, population sizes between 20 and 50 design space points are common (Heredia-Langner (2001)). Mayer et al. (2001) recommends a population size greater than the dimensionality, the number of factors, of the studied problem and prefers using a population size equal to twice the dimensionality. They also warn of potential dangers in choosing too small of a population.

Parent/Offspring Ratio

The parent/offspring ratio determines the number of offspring design space points to create from the previous parent populations to produce the new parent population. For example, a parent-to-offspring ratio of 1:7 from a parent population of 50 implies that 350 offspring will be produced. The overall desirability for each of these 350 offspring will be calculated. These offspring individuals will then undergo some sort of selection procedure to determine which will become members of the next parent population.

Bäck (1996) stated that, for every problem, there is an optimal parent-to-offspring ratio that yields the most reasonable compromise between computational effort and progress-rate gained. Thus, we will set this parameter with ratios of 1:1 and 1:7 for the designed experiment.

Selection Type

The offspring created in each GA generation are evaluated and ranked so that better performing design space points are selected for use in future generations. Several selection procedures are available to consider, including ranking selection, proportionate reproduction, and tournament selection. Tournament selection chooses a previously determined number of offspring individuals, or competitors, (at random and with replacement), evaluates them, and then selects a few of the best ones to become members of the new parent population. This process is repeated until all the spaces available for the parent population have been filled. Reed et al. (2000) indicates that tournament selection is robust across problem types. Tournament selection allows poorer performing offspring some chance to survive. The performance of tournament selection depends on the number of competitors specified. As the number of competitors decreases, selection pressure and convergence rates decrease too. To compensate, we set the number of competitors equal to twice the selected parent-to-offspring ratio. For example, with a parent-to-offspring ratio = 1:7 and parent population = 50, the tournament selection pool consists of $2 \times 7 = 14$ of the total 350 offspring created.

Crossover Rate

The crossover rate refers to the percent of the parent population that will undergo a crossover operation. A wide range of crossover rate values has been examined (Mayer et al. (2001)). A single-point crossover technique is employed throughout our experimentation because the chromosome string length is short. While a high crossover rate typically causes good points to be discarded, a low crossover rate places too much emphasis on parents and may stagnate the search. For this experiment, the crossover rate settings are based on values suggested by Wehrens et al. (1999), with low and high values of 0.5 and 0.85, respectively.

Mutation Type and Rate

Mutation refers to the altering of one or more x_i (genes) in $\mathbf{x}$ (a chromosome). Mutation types are characterized by how many x_i are altered and by the degree to which x_i is changed. Two types appropriate for real-valued coding are uniform and Gaussian mutation. Uniform mutation changes only one x_i according to a uniform distribution. The Gaussian type changes all the x_i by a small amount according to a normally distributed disturbance.

A chromosome altered by Gaussian mutation results in a point located in the neighborhood of its parent. Gaussian mutation alters the vector $\mathbf{x}$ by adding a randomly generated vector $\mathbf{m} = (m_1, m_2, \ldots, m_k)$ consisting of mild, normally distributed perturbations. The new design space point becomes $\mathbf{x}^* = \mathbf{x} + \mathbf{m}$. Both uniform and Gaussian mutation types are considered in this experiment.

For real-value coding, Bäck (1996) suggests using mutation rates as high as 0.6. In this experiment, mutation rates of 0.1 and 0.4 are considered.

Genetic Algorithm Performance Study: Noise Factors and Problem Complexity

In this section, factors related to problem complexity will be discussed. There are several attributes that can be used to quantify or characterize the complexity of a problem. For example, one measure of problem complexity is the number of input variables in individual response surface models. Altering the complexity of various problem attributes can result in a test bed of case studies for evaluation. The following section describes four problem attributes investigated in an experiment. Varying the problem attributes will generate problem case studies for evaluation.

Number of Factors

The number of factors determines the dimension of the search or design space as well as the length of $\mathbf{x}$. The value set for a particular scenario represents the total number of factors that will serve as candidates for random selection in each modeled response. In an example later, we will see that each modeled response will randomly select a subset from the factors set. Selection for the low- and high-level settings of this attribute is based on the number of significant factors typically discovered in industrial designed experiments. A low-level setting of four factors was chosen. This setting will produce cases of moderate size such that traditional optimization techniques should perform well. The high-level setting of eight variables is selected to reflect a value that occurs in practice yet is large enough to challenge the common optimization methods.

Number of Responses

The number of responses represents the number of performance measures simultaneously monitored. This factor does not add to the dimension of the problem, but its value will affect the complexity of the feasible region and shape of the overall desirability function. A low-level setting of four responses will produce cases of moderate size that traditional optimization techniques should still be capable of solving. In response surface settings, it is not unusual to find applications with 20 or more response variables (Montgomery (1999)). Therefore, we chose a high-level setting of 16 responses, indicative of moderate- to large-sized problems.

Constraints Range

The constraint range for each response indicates half the width as a percentage of the response magnitude of the interval between the lower and upper bound of $d_i(\hat{y}_i)$. This parameter can drastically affect the size of the feasible region. Industrial experience and a desire to challenge solution methods' performance influenced the low- and high-level settings of 5% and 15%. An example 5% constraint range for a target response value of 100 would yield lower and upper desirability limits of 95 and 105, respectively.

Percent of Second-Order Response Models

The choices for the order of a particular response model in this evaluation are either linear plus interaction or complete second order, which includes pure quadratic terms. This problem attribute represents the ratio of complete second-order response models to the total number of response models in a particular multiple-response problem. The percent of second-order models attribute will most likely contribute to the shape of the overall desirability function. The low- and high-level settings were chosen to be 25% and 75%.

Other Problem-Complexity Decisions

Decisions are necessary for other problem attributes associated with building multiple-response surface models. Certain rules were used to ensure that the problems generated mimic problems typically found in industry (Table 2).

TABLE 2. Individual Response Model Generation Decisions

Factors are selected randomly for each response.
For linear plus interaction models, 1/2 of the terms are linear (main effects), 1/2 of the terms are two-factor interactions.
For second-order models, 1/2 of the terms are linear (main effects), 1/4 of the terms are interactions, 1/4 of the terms are pure quadratic.
The regression coefficient magnitudes and sign are randomly selected from a uniform distribution $U(5, 20)$ and the model intercept equals 50.
Model hierarchy is always maintained. Therefore, any variable present in an interaction or pure quadratic term is also included as a main effect.
A value of 1 is used to weight variables s_i and t_i in Equation (2).

To illustrate the above procedures, consider a situation with four factors and four responses, the percent of second-order models is 75%, and a 15% constraint range yields lower- and upper-constraint bounds of 42.5 and 57.5, respectively. Following the rules presented, the following four regression models represent a possible outcome of the simulation:

$$\hat{y}_1 = 50 + 16x_1 - 6x_2 + 12x_1x_2 + 7x_1^2$$
$$\hat{y}_2 = 50 - 19x_2 + 8x_3 + 13x_2x_3$$
$$\hat{y}_3 = 50 + 19x_1 + 18x_2 - 8x_1x_2 + 11x_3^2$$
$$\hat{y}_4 = 50 - 7x_3 - 19x_4 + 10x_3x_4 - 14x_4^2.$$

Each response will have a target value of 50, the same as the intercept. Thus, the coded factors vector $\mathbf{x}$ that produces the optimum solution $D(\mathbf{x}) = D^*(\mathbf{x}) = 1.0$ is $\mathbf{x} = [0, 0, 0, 0]$. This relationship is established to ensure that the optimum solution for any problem is known and control is maintained over the complexity of the problem.

The $D(\mathbf{x})$ examined in the simulation can be more complex than the response surface shown in Figure 1. Each response is a two-dimensional or greater problem. The number of responses ranges from 4 to 16 and the number of factors among these responses ranges from 4 to 8. Imagine $D(\mathbf{x})$ as a series of high-dimensional contour plots overlapping each other resulting in a highly nonlinear region. Only small regions will contain feasible solutions. This type of problem will cause some conventional deterministic optimization algorithms trouble because a starting solution close to (or within) a feasible region would be required.

Genetic Algorithm Performance Measures

The performance of the genetic algorithm is measured in terms of the speed of finding a feasible solution and the rate at which a high-quality solution is obtained. Finding feasible solutions is often a significant challenge for multiple-response optimization techniques. One indicator of success for the GA is the number of evaluations performed until a feasible solution is found. An evaluation refers to a $\mathbf{x}$ examined for its corresponding overall desirability value. Hence, this performance measure calculates every search point examined until a feasible solution is found. The second GA performance measure is the effort required to obtain a high-quality solution. This response counts the number of evaluations until the best solution is within 10% of the known optimal. Knowing the target $D(\mathbf{x}) = 1.0$, the criteria counts the number of $\mathbf{x}$ examined until $D(\mathbf{x}) \geq 0.9$. These two performance measures will serve as the responses in a designed experiment and analysis of the GA parameter settings.

Robust Genetic Algorithm Performance Investigation

A robust design approach is used to evaluate the performance of the GA with different parameter settings against a wide range of multiple-response problem scenarios that vary in complexity. This study will evaluate six GA parameters and four problem-complexity attributes (Table 3). The GA parameters represent the control variables in a robust parameter design while the problem-complexity attributes represent noise variables. We use a combined array strategy that assimilates control and noise factors into a single fractional factorial design

A 2_{IV}^{10-4} fractional factorial design is selected consisting of 64 runs. This resolution IV design is nearly resolution V, as the majority of the two-factor interactions are aliased with three- and four-factor interactions. Some two-factor interactions are aliased

TABLE 3. Levels of the Genetic Algorithm Control and Noise Parameter Setting

Factor	Parameters	Low	High	Variable Type
A	Parent population	20	50	Control
B	Parent/offspring ratio	1:1	1:7	Control
C	Selection type	Rank	Tournament	Control
D	Crossover rate	0.5	0.85	Control
E	Mutation type	Uniform	Gaussian	Control
F	Mutation rate	0.1	0.4	Control
G	Number of factors	4	8	Noise
H	Number of responses	4	16	Noise
J	Percent of second-order response models	25	75	Noise
K	Constraint width (% of target)	5	15	Noise

with other two-factor interactions, so further experimentation may be necessary to identify which term is significant. Four center points are also added to provide a pure error estimate and a test for curvature. Because two control variables are categorical, pseudo-center points are used such that centers of the numeric factors are run for every combination of the categorical factors. Two categorical factors at two levels each produce $2^2 = 4$ pseudo-center point combinations. Four replicates of each pseudo-center point combination require 16 runs to be added to the original 64, equaling 80 total experiments.

The focus of the experiment analysis is on building empirical models for GA performance containing control variables (genetic algorithm parameters) and noise variables (problem-complexity attributes). Model building involves selecting terms according to a robust design response surface model of the form

$$\hat{y}(\mathbf{x},\mathbf{z}) = b_0 + \sum_{i=1}^{6} b_i x_i + \sum_{i=1}^{6} \sum_{j=1, j\neq i}^{6} b_{ij} x_i x_j + \sum_{i=1}^{4} c_i z_i + \sum_{i=1}^{6} \sum_{j=1}^{4} d_{ij} x_i z_j,$$

where the x_i represent the GA tuning-parameter control variables and the z_i refer to the problem-complexity noise variables. Interpretation clearly involves a careful study of the control main effects, control × control interactions, and control × noise interactions.

The data for both performance measures require variance-stabilizing transformations to satisfy model assumptions. For both performance measures, a natural log transformation proved most effective. The response surface models for both GA performance measures included only significant noise main effects, control × control interactions, and control × noise interactions. The significant noise main effects (G, H, and K in both models) essentially indicate that the problem-complexity attribute levels selected for this experiment do impact GA performance, a desired outcome in this study. The control × control and control × noise interactions provide the answers for how to best tune the GA. In fact, interaction plots clearly reveal the appropriate genetic algorithm parameter settings needed to optimize each GA performance measure across the spectrum of problem complexities. A summary of the findings is provided in Table 4.

Based on the findings in Table 4, the control variables are then set to maximize performance, which equates to minimizing the number of evaluations to achieve each objective (Table 5). Nearly all guidance is clear and in agreement. Only factor B, the parent population to offspring ratio, is in apparent disagreement. Closer inspection shows that the 1:1 ratio is only recommended to achieve quicker feasibility for relatively easy problems. Thus, the 1:7 ratio, which enables faster achievement of $D(\mathbf{x}) > 0.9$ in challenging scenarios, is selected. The table also provides guidance for all control variables except factor D, the crossover rate. Higher crossover rates assist in diversifying the offspring pool, so that parameter is set to 0.85.

The Robust Genetic Algorithm as an Alternative to the GRG

The robust genetic algorithm should ultimately be compared with a multiple-response desirability

TABLE 4. Results of the GA Performance Investigation Robust Parameter Designed Experiment

GA performance measure: *Achieve feasibility*		
Significant term	Term type	Finding
BH	Control × noise	A 1:1 parent/offspring ratio performs better with only four responses but the offspring ratio does not matter with a moderate or large number of responses.
BK	Control × noise	A 1:1 parent/offspring ratio reaches feasibility more quickly for wider (15%) constraints and this ratio does not influence feasibility performance with moderate to tight (5%) constraints.

GA performance measure: *Achieve* $D(\mathbf{x}) \geq 0.90$		
Significant term	Term type	Finding
AC	Control × control	For parent population = 20, selection type does not matter, whereas if the parent population = 50, choose tournament selection.
AE	Control × control	Better performance is achieved with parent population = 20 and Gaussian mutation.
AF	control × control	Performance is enhanced using the parent population = 20 and a mutation rate = 0.4.
BK	Control × noise	An offspring ratio of 1:7 is best with tight constraints (5%), but as the constraints widen, the offspring ratio is unimportant.
EG	Control × noise	Gaussian mutation works better with only 4 factors and both mutation methods perform equally with 8 factors.
EK	Control × noise	Gaussian performs better with tight constraints (5%), but as the constraints widen, the mutation type is insignificant.
AEK	Control × noise	For mutation rates of 0.1, use a parent population = 20 and Gaussian mutation. For all other mutation rates, factor AE is not significant.
CFK	Control × noise	For tournament selection, use mutation rate = 0.4 regardless of constraint setting. For ranking selection, the FK interaction is not significant.

TABLE 5. Final Settings for the Proposed Robust Genetic Algorithm

Factor	Parameters	Setting	Rationale
A	Parent population	20	Designed experiment result
B	Parent/offspring ratio	1:7	Designed experiment result
C	Selection type	Tournament	Designed experiment result
D	Crossover rate	0.85	Diversify offspring pool
E	Mutation type	Gaussian	Designed experiment result
F	Mutation rate	0.4	Designed experiment result

method that is widely available and widely used in practical applications. The method by Del Castillo, Montgomery, and McCarville (1996) using a modified desirability function and generalized reduced gradient (GRG) optimization is selected because the GRG is perhaps one of the most popular and robust nonlinear programming methods available (Del Castillo and Montgomery (1993)). The GRG is also preferred for optimizing multiple-response problems using a loss function objective (Vining (1998)).

The GRG belongs to a class of optimization techniques known as gradient-based methods. Like direct search methods, the GRG starts off at a trial point and tries to find a direction of movement that will improve the objective function. Unlike direct search methods, the GRG does require the calculation of derivatives to solve a problem. Due to this, the GRG is more efficient than direct search methods. However, the GRG does experience difficulties solving problems if the starting point is far from optimum and if the constraints are highly nonlinear (Gill et al. (1981)).

GRG Optimization Algorithm Performance Investigation

The GRG performance will be evaluated against the problem-complexity scenarios generated for the robust design. Of the four problem-complexity attributes introduced in the previous section, only three affected the performance of the genetic algorithm. The percent of second-order response models was not significant, leaving the three attributes in Table 6.

A 2^3 factorial designed experiment with one center run results in nine problem scenarios forming the case studies for evaluation (Table 7). The same approach described in the last section to generate specific response models for each scenario was used in this evaluation. The percent of second-order models was set to the more challenging level (75%) from the robust design.

The Del Castillo, Montgomery, and McCarville approach is coded in Excel to take advantage of Excel's Solver, which performs GRG optimization. The performance of gradient-based methods such as the GRG is enhanced by multiple starting points. Thus, the GRG method is applied to each of the nine cases using 30 randomly selected starting locations. Results are recorded for each starting point.

TABLE 6. Levels of Problem Parameters Settings

Factor	Problem Parameters	Low	High
A	Number of factors	4	8
B	Number of responses	4	16
C	Constraints (% of target)	5%	15%

TABLE 7. Design Matrix and the Collected Data

Case no.	Factor A no. of factors	Factor B no. of responses	Factor C constraint range
1	4	4	5
2	8	4	5
3	4	16	5
4	8	16	5
5	4	4	15
6	8	4	15
7	4	16	15
8	8	16	15
9	6	10	10

GRG performance is measured by calculating the percentage of times that the GRG found the optimal solution ($D(\mathbf{x}) > 0.9$) out of the 30 randomly chosen starting points. Table 8 reports the experiment results. The high levels of the case scenario settings are bolded to enable easier identification of the more challenging scenarios.

The table clearly shows that the performance of the GRG deteriorates with increasing scenario complexity. For the relatively easy scenarios (Cases 1, 5, 6, and 7), the GRG performed reasonably well, finding the optimal solution between 27 and 83% of the time. For moderately challenging scenarios (Cases 2, 3, and 9), the optimal solution is found in only 2 of the 30 initial starting points. For the more challenging scenarios (Cases 4 and 8), no feasible solution is found.

Comparison Between GA and GRG Optimization

To evaluate and compare the performance of the GA desirability function versus the GRG algorithm, the GA desirability function is applied to the same nine cases used for the GRG analysis (Table 9). The results show the performance of GA to be robust regardless of problem complexity. Here "robust" refers

TABLE 8. Performance Study of the GRG Method Indicating the Percentage of Times the Optimal Solution is Obtained

Case	No. of factors	No. of responses	Constraint range	Pct of GRG solutions optimal
1	4	4	5	0.30
2	8	4	5	0.07
3	4	16	5	0.07
4	8	16	5	0.00
5	4	4	15	0.83
6	8	4	15	0.27
7	4	16	15	0.57
8	8	16	15	0.00
9	6	10	10	0.07

TABLE 9. Performance of Proposed Genetic Algorithm Against the Multiple-Response Cases

Case	No. of variables	No. of responses	Constraints	$D^*(\mathbf{x})$ mean	St. dev.	(95.0%) Confidence intervals	
						High	Low
1	4	4	5	0.989	0.004	0.994	0.985
2	8	4	5	0.969	0.011	0.979	0.958
3	4	16	5	0.997	0.002	0.998	0.995
4	8	16	5	0.987	0.005	0.991	0.982
5	4	4	15	0.998	0.001	0.998	0.997
6	8	4	15	0.994	0.003	0.997	0.992
7	4	16	15	0.987	0.004	0.991	0.983
8	8	16	15	0.958	0.027	0.984	0.931
9	6	10	10	0.979	0.016	0.995	0.962

to the GA being able to solve all cases with a high level of quality and consistency. The final solutions obtained have mean values ranging from 0.958 to 0.998 (out of a possible 1.0), indicating the GA final solutions are very close to the optimal solution. Optimization was performed four times for each design point. The modest sample size and corresponding tight confidence interval widths indicate that the GA is also reproducible.

Example: Optimization of a Multiple Response Lexmark Toner Optimization Problem

To illustrate the approach, we present an example of a Lexmark toner study. Toner is a composite of materials created by melt mixing bulk ingredients in an extruder and milling the results into to a small particle medium, which is the consistency of talc. The properties of the resulting composite enable it to be moved in a controlled manner and then affixed to a print medium to create an image. The purpose of this particular experiment is to determine the effect of changing three toner additives on various toner performance metrics, which measure the quality of the printed image and the efficiency of toner usage.

For proprietary reasons, the names of the additives are not given, but the response variable names are provided. The factor levels used are shown in Table 10. The factor levels indicate the percent of total. For example, the additive 1 low level $(-1) = 0.20\%$. Because these additives represent only a small portion of the overall mixture, a standard factorial approach is reasonable.

A central composite design was used with a total of 20 experimental runs. Response surface model fit-

TABLE 10. Factor and Levels

Factor	Name	Units	Low level (−1)	High level (1)
A	Additive 1	%	0.2	0.5
B	Additive 2	%	0.5	1
C	Additive 3	%	0.2	0.5

ting was applied to the data to obtain appropriate regression models for each of the 14 responses. The results are summarized in Table 11.

A company chemical engineer set objectives for each response. A summary of the goals for each response as well as the minimum and maximum values for the responses and their associated desirability values are provide in Table 12.

The engineer chose to restrict or constrain the de-

TABLE 11. Summary of Design Experiment Results

Response	Name	Units	Minimum value obtained	Maximum value obtained	Model chosen
Y_1	Streak uniformity rank	Rank	1	20	2FI
Y_2	Streak uniformity rating	Rating	1	6	2FI
Y_3	Speckles	Rating	0	3	Quadratic
Y_4	Starve	Rating	0	3	Quadratic
Y_5	Mottle	Rating	1	19	Linear
Y_6	Powder flow	%	13.2	25.5	Linear
Y_7	Developer roll mass/area delta	mg/cm^2	−0.45	1.43	Quadratic
Y_8	Developer roll charge/mass delta	μC/g	−9.3	11.6	Quadratic
Y_9	Photoconductor charge/mass delta	μC/g	−11.7	18.7	Quadratic
Y_{10}	Photoconductor mass/area delta	mg/cm^2	0.54	1.24	Quadratic
Y_{11}	Toner to cleaner average	mg/page	3.3	11	Quadratic
Y_{12}	Toner to cleaner at 10K pages	mg/page	2.6	9.8	Quadratic
Y_{13}	Solid fill optical density	density	1.28	1.86	Quadratic
Y_{14}	Film onset	pages	10	25	Quadratic

TABLE 12. Goals and Constraints for Each Response

Response	Goal	$Y_{\min}$	$Y_{\max}$	$d_i(Y_{\min})$	$d_i(Y_{\max})$
Y_1	Minimize	1	20	1	0
Y_2	Minimize	1	6	1	0
Y_3	Minimize	0	3	1	0
Y_4	Minimize	0	3	1	0
Y_5	Minimize	1	19	1	0
Y_6	Mmaximize	13.2	25.5	0	1
Y_7	Minimize	−0.45	0.2	1	0
Y_8	Minimize	−9.3	11.6	1	0
Y_9	Minimize	−11.7	18.7	1	0
Y_{10}	Minimize	0.54	1.24	1	0
Y_{11}	Minimize	3.3	5	1	0
Y_{12}	Minimize	2.6	5	1	0
Y_{13}	Is in range	1.3	1.6	1	1
Y_{14}	Maximize	10	25	0	1

TABLE 13. Response Value Associated with the Best Solution

Response	Results of optimal settings	Goal	$Y_{\min}$	$Y_{\max}$
Y_1	10.54	Minimize	1	20
Y_2	3.70	Minimize	1	6
Y_3	0.19	Minimize	0	3
Y_4	2.24	Minimize	0	3
Y_5	7.67	Minimize	1	19
Y_6	17.45	Maximize	13.2	25.5
Y_7	0.18	Minimize	−0.45	0.2
Y_8	−2.12	Minimize	−9.3	11.6
Y_9	−2.61	Minimize	−11.7	18.7
Y_{10}	0.70	Minimize	0.54	1.24
Y_{11}	7.19	Minimize	3.3	5
Y_{12}	3.58	Minimize	2.6	5
Y_{13}	1.46	Is in range	1.3	1.6
Y_{14}	24.28	Maximize	10	30

sirability for four responses: Y_7, Y_{11}, Y_{12}, and Y_{13}. For the Developer Roll Mass/Area Delta response (Y_7), the engineer prefers a delta less than 0.2. For both the Toner to Cleaner Average (Y_{11}) and Toner to Cleaner at 10K Pages response, (Y_{12}), the engineer is concerned about values less than 5 mg/page. The Solid Fill Optical Density (Y_{13}) response should be in the range of about 1.3 to 1.6.

Optimization evaluations (using both the GA and the GRG) revealed that no $\mathbf{x}$ setting exists that will satisfy all response constraints simultaneously. As such, the engineer wanted to know which response(s) is (are) not feasible and how close the best solutions are to feasibility. Because the proposed multiple-response method uses an unconstrained desirability function, it is capable of finding the solutions most nearly satisfying all constraints. The robust GA parameter settings are used to set up the GA for this example. Fifty generations were performed. The best solution found is $\mathbf{x} = (-0.155, 0.023, 0.857)$. The corresponding response values for this setting of additives are shown in Table 13.

Using the GA in conjunction with the unconstrained desirability function reveals that only the constraint for Y_{11} is violated. The engineer can use this information to examine response Y_{11}. If the regression model used for Y_{11} is accurate and significant, the practitioner should reexamine the goal and the constraints for this response. If it is found that the goal and the constraints for this response are vital, the engineer must conduct further studies to determine what can be done to bring Y_{11} to an acceptable level. If the goal and the constraints for this response are not vital, the engineer can choose to change the goal for this response or remove it entirely from the study. Afterwards, another attempt at optimizing the problem can then be attempted.

This example demonstrates one of the key strengths of using the proposed optimization technique. The proposed technique is able to rank order design space locations $\mathbf{x}$ based on their degree of infeasibility. The GA can then use this information to guide its exploration for feasible points. Clearly, the example could grow more challenging quickly by increasing the number of responses with bounded desirabilities. The advantage of the proposed method in assessing the design space fairly exhaustively would probably be even more evident with problems of increased complexity, where conventional optimization methods may examine only points in the immediate vicinity of starting locations and fail to converge to high-quality solutions.

Conclusion and Recommendations

This paper proposes a technique for solving multiple-response problems in response surface applications. The approach consists of an unconstrained desirability function combined with a genetic algorithm. The commonly used Derringer and Suich overall desirability function possesses characteristics that

hinder optimization performance in a genetic algorithm framework. The geometric mean of the individual desirabilities prevents the genetic algorithm from comparing overall desirability values of infeasible $\mathbf{x}$. The proposed unconstrained desirability function is suited for the GA by enabling the algorithm to differentiate between far-from-feasible and nearly feasible solutions. Delineation is accomplished by incorporating a penalty function proportional to the magnitude of the constraint violation into the overall desirability value of each design space point.

A drawback of the genetic algorithm is the tuning required to properly select parameters, including the parent population size, the ratio of initial parent population to offspring size, the selection technique, the crossover rate, the mutation type, and the mutation rate. The performance of the GA is dependent on the parameters chosen. A robust parameter designed experiment was performed to determine the parameter settings that performed the best across a variety of multiple-response problems. The resulting method was then compared with a widely used desirability function alternative, the GRG. The performance study shows that, while the GRG struggles with more complex multiple-response problems, the proposed GA technique consistently solves problems regardless of their complexity.

Using the GA for less complicated multiple-response problems can also be beneficial because the genetic algorithm explores a larger portion of the design space searching for potential optimum solutions. A practitioner may find that the optimal solution lies in too small an operating region and wish to find a larger operating region that will produce close-to-optimal performance. The points investigated by the GA in generations prior to convergence can assist in mapping the entire solution space and identifying these promising regions that produce favorable results and allow flexibility among the factor settings. This information is not easily accessible with conventional direct or gradient-based search methods.

Finally, the proposed approach to multiple-response optimization can be easily modified to incorporate loss or distance functions that can include the important aspect of correlations among the responses. These distance or loss functions would just replace the desirability function as the fitness in the genetic algorithm. The approach detailed in this paper for modifying the objective function and tuning the GA could be appropriately applied.

Appendix: An Overview of Genetic Algorithms

Genetic algorithm (GA) is a term used for a search technique that incorporates the concepts of natural selection in its iterative steps. GAs use historical information from previously examined solutions in selecting new search points where improved performances are expected. GAs differs from conventional optimization algorithms in that they examine a population of points at each iteration rather than one point and they use the objective function (fitness function in GA terminology) rather than the derivative or gradient directly in the search.

Throughout its run, the GA maintains a population pool of potential solutions, called a generation, where each potential solution is referred to as chromosomes. The GA starts off by randomly generating a population of chromosomes that represents the initial mating pool. Generally, a larger initial population increases the potential to explore the solution space thoroughly; however, it also increases computation time and slows down convergence. The GA then evaluates the quality of each individual chromosome using a fitness function. Once the fitness of each chromosome is evaluated, a selection process takes place to determine which chromosomes will be used as the parents of the next generation. The GA uses various evolving operations to create a new generation of chromosomes (called offspring) that hopefully will take on the positive characteristics of their parents and thus produce near-optimum solutions. Evolving operations are categorized in two different classes: recombination, and mutation. Future generations are created performing selection, recombination, and mutation iteratively until a stopping criterion has been met. Some commonly used stopping criteria are to terminate after a predetermined number of iterations, when there has been no change in the fitness value of the best solution after a predetermined number of iterations, or when a large percentage of the chromosomes in the population pool are the same.

The flow chart in Figure A.1 illustrates one way the steps of the GA can be implemented.

The following sections will provide a brief explanation of the steps and procedures often used in the genetic algorithm. A more complete description can be found in Goldberg (1989) and Herendia-Langner et al. (2003).

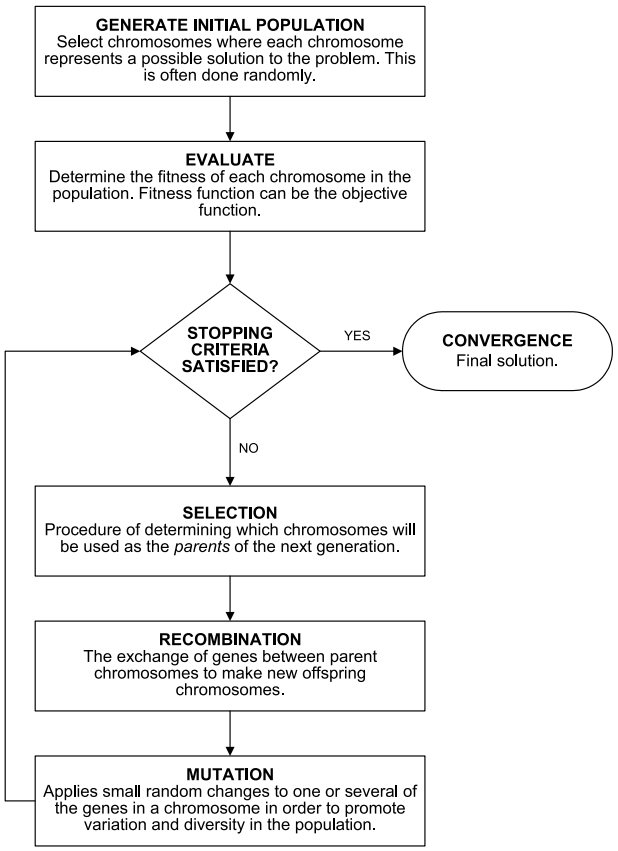


FIGURE A.1. Genetic Algorithm Flow Chart.

Coding

The initial step in the implementation of the genetic algorithm is the encoding of potential solutions. The most widely used method for encoding solutions is to represent every solution as a binary $(0, 1)$ string. For example, consider a response surface empirical model that serves as an objective function for the GA,

$$\hat{y} = 10x_1^2 + 5x_1x_2 - 25x_2^2.$$

A potential solution can be represented as a binary string, where the first half of the string represents the value for x_1 and the other the value for x_2. Each $(0, 1)$ value in the chromosome is referred to as a gene. For example, the chromosome

0	0	0	0	1	0	1	0	0	0	0	1	1	0	0	0	0	0	0	0

is partitioned into two halves

0	0	0	0	1	0	1	0	0	0

and

0	1	1	0	0	0	0	0	0	0

these strings are then converted from base 2 to base 10 to yield

$$x_1 = 40 \quad \text{and} \quad x_2 = 384.$$

The length of a chromosome increases as the number and magnitude of decision variables increases. Due to this structure, the computing resources needed to run the GA using binary encoding can make this procedure inefficient. An alternative to binary coding is to use real-value entries for the genes. The following example uses real-value coding. Consider a response surface empirical model,

$$\hat{y} = 174.93 + 23.38x_1 + 3.62x_3 - 19.00x_2x_3.$$

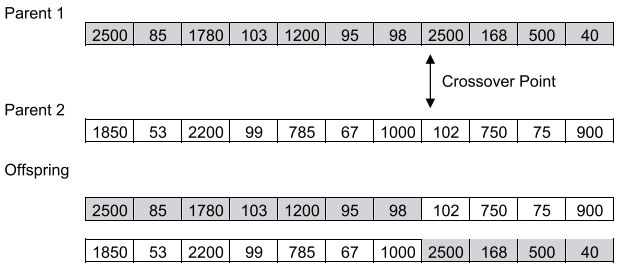
A chromosome in this problem could be a three-dimensional vector where each entry represents the value (in coded units) for each decision variable.

0.125	-1.000	0.525
x_1	x_2	x_3

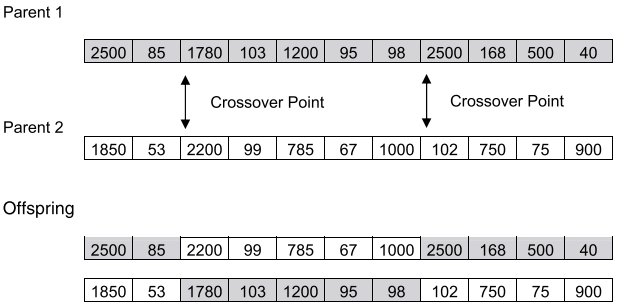
The coding is very simple to understand in this case. While the procedure for recombination and mutation differ slightly, the basic concepts and difficulty remain the same.

Recombination

Recombination refers to the exchange of genes between parent chromosomes to make new offspring that hopefully will take on the positive characteristics of their parents. The most common recombination operation is called the crossover. The crossover operation randomly takes a pair of chromosomes (parents), splits the chromosomes at the same positions (the crossing sites), and creates an offspring chromosome by combining the alternate portions of the parent individuals. The following is an example of a one-point crossover.



Multiple crossover points are implemented in the same way:



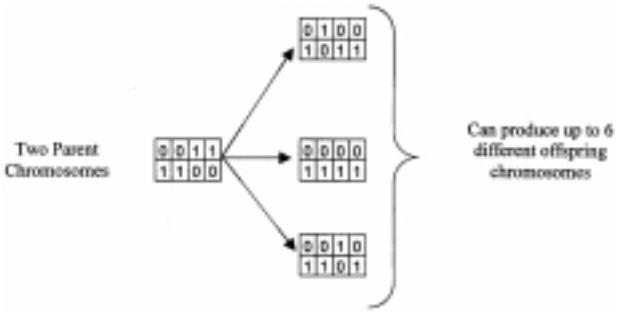


FIGURE A.2. All Possible Combinations from One Point Crossover.

There are many other recombination techniques that can be used, such as using more than two parents in the crossover procedure. Another operation, called generalized intermediate recombination, makes each entry of the offspring chromosome a combination of both parents rather than a copy of just one. For example, in the previous illustration, the first entry from parents 1 and 2 are 2500 and 1850, respectively; the offspring entry could simply be the average of the two, 2175.

The GA cannot rely on recombination alone to search the entire solution space efficiently. Doing so will cause the genetic algorithm never to venture outside the search space created by the initial population and potentially cause the algorithm to get trapped in a local optimum (Heredia-Langner (2001)). To prevent premature convergence, the GA generally uses an evolving operation known as mutation.

Parent/Offspring Ratio

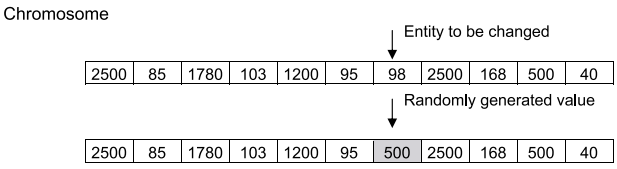
The purpose of the recombination procedure is to help explore the entire search space thoroughly. Thorough exploration can successfully be done if you allow each parent to interact with other chromosomes as many times as is computationally possible (see Figure A.2).

Using single-point crossover, two parent chromosomes with string lengths of four can be recombined to produce a total of six different offspring chromosomes. If the mating process is set up to only allow a parent chromosome to interact once during the mating process, the large potential knowledge that would have been gained from allowing that parent to interact with potentially better parents is lost. It would be ideal to let each parent interact with every other chromosome in every possible combination. Unfortunately, due to the length of chromosome strings and the large number of parents in a population, allowing parents to interact this extensively is computationally expensive and sometimes impossible. A technique originally introduced by Schwefel (1997) is to generate an offspring pool size that is a multiple of the parent pool size. Schwefel has recommended ratios of 1:5 or 1:6 to be used. There is an optimal parent/offspring ratio that yields the most reasonable compromise between computational effort to be invested and progress rate gained for every problem (Bäck, 1996).

Mutation

Mutation applies small random changes to one or several of the genes in a chromosome in order to promote variation and diversity in the population. Mutation helps widen the search space and explore more of the response surface. Several mutation procedures have been proposed in the literature, and three common mutation operations are discussed in this paper: uniform, multiple uniform mutation, and Gaussian mutation.

The uniform mutation operation randomly selects a gene from the parent chromosome and replaces it with a random number within the range of the solution space. The following example illustrates this process:



The multiple uniform mutation operation follows the same methodology as the uniform mutation operation except that it is applied to several of the genes in a chromosome. The number of genes selected for mutation is chosen randomly.

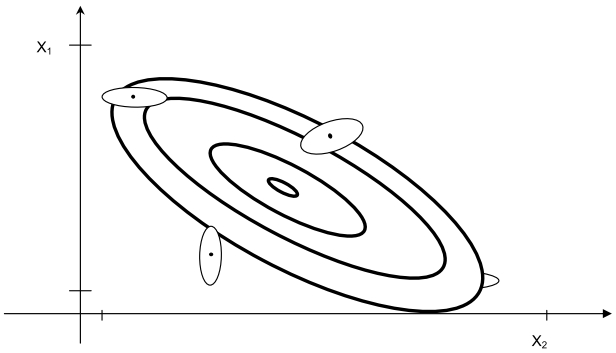


FIGURE A.3. Gaussian Mutation on a Hypothetical Objective function.

The Gaussian mutation operation slightly changes all genes of the chromosome such that the resulting chromosome lies somewhere near the neighborhood of its parent. The Gaussian mutation operation helps the GA converge faster to a solution but also increases the chance of stopping at a local optimum. A general description will be provided here. A full description can be found in Bäck (1996).

The changes performed to the genes are normally distributed with mean zero and standard deviation σ_i, where i is the ith gene in a chromosome of length l. The setting for the standard deviation, σ_i, is determined by the user and allows the user to define the appropriate magnitude of the change for every gene involved. Gaussian mutation places the offspring in the hyperellipsoids of constant probability defined by the multivariate normal distribution specified. Figure A.3 is from Heredia-Langner et al. (2003) and illustrates the effect Gaussian mutation has on the affected chromosomes. The thick ellipses represent contours of equal objective value. The dots at the center of the small ellipses are the original chromosomes before mutation. The offspring chromosomes will be located somewhere in the normally distributed neighborhood, indicated here by the small ellipsoids.

Selection

Once new offspring chromosomes are created by the evolving operations described in the previous sections, a procedure must then be implemented to determine which chromosome will be used as the parents of the following generation. As with recombination and mutation, there are several procedures to consider. The following is a brief explanation of some of those procedures.

Proportionate reproduction is a term used to describe several different schemes that choose individuals for reproduction due to their fitness value. The general procedure starts by ordering the population according to their fitness and then by using some biased probabilistic method. The chromosome with better fitness values stands a better chance of being selected for reproduction than those with poor fitness values. This method gives every chromosome in the population a chance to reproduce even if it has a poor fitness value. The argument is that even a poor chromosome can have good genes.

Ranking selection is a deterministic process as opposed to the probabilistic process of proportionate reproduction. In this scheme, a population is sorted from best to worst until the desired number of parents is reached. Only the top chromosomes get to reproduce despite the fact that some poor performing chromosome may possess good genes.

The tournament selection procedure is a combination of the previous two described methods. In this procedure, the population is shuffled into groups of chromosomes randomly. Then the best individuals are chosen from each group. This procedure is repeated until the required number of parents is reached.

Replacement Strategy

There are two widely used methodologies for replacing the parent pool with newly created offspring, generation gap and steady state. Generation gap replacement requires all parent chromosomes to be replaced by the next generation of offspring. Most early uses of the genetic algorithm used this strategy. Typically, an elitist strategy (where the best parent is kept) is implemented in conjunction with the generation gap methodology. More recently, steady-state replacement has become more popular (Mayer (2001)). Under the steady-state replacement strategy, offspring chromosomes are compared with the parent pool right after creation and the poorest performing chromosomes are driven to extinction immediately. Hence, only the best chromosomes survive and good performing offspring chromosomes are immediately available for reproduction.

Concluding Remarks on Genetic Algorithms

Determining the appropriate operators and parameter settings to use is key to the GA optimization performance. The proper choices for operators and parameter settings are problem specific. A problem that is highly nonlinear and heavily constrained

can benefit from GA parameter settings that promote more exploration and less convergence, thus improving the likelihood of finding a feasible solution and minimizing the chances of converging to a suboptimal solution. A problem that is not as complex should use GA parameter settings that promote convergence and allow the GA to settle quickly to the best solution. Using a robust parameter design approach can be useful in determining the proper parameter settings that perform the best across a variety of problems.

Acknowledgements

The authors would like to thank the editor and the two anonymous referees for their helpful suggestions on an earlier draft of this paper.

References

Bäck, Thomas. (1996). *Evolutionary Algorithms in Theory and Practice.* Oxford University Press, Oxford, New York.

Borkowski, J. J. (2003). "Using A Genetic Algorithm to Generate Small Exact Response Surface Designs". *Journal of Probability and Statistical Science* 1(1), pp. 65–88.

Carlyle W. M.; Montgomery, D. C.; and Runger, G. C. (2000). "Optimization Problems and Methods in Quality Control and Improvement". *Journal of Quality Technology* 32, pp. 1–17.

Del Castillo, E. and Montgomery, D. C. (1993). "A Nonlinear Programming Solution to the Dual Response Problem". *Journal of Quality Technology* 25, pp. 199–204.

Del Castillo, E.; Montgomery, D. C.; and McCarville, D. R. (1996). "Modified Desirability Functions for Multiple Response Optimization". *Journal of Quality Technology* 28, pp. 337–345.

Derringer, G. and Suich, R. (1980). "Simultaneous Optimization of Several Response Variables". *Journal of Quality Technology,* 12, pp. 214–219.

Gill, P. E.; Murray, W.; and Wright, M. A. (1981). *Practical Optimization.* Academic Press, London, England.

Goldberg, D. E. (1989). *Genetic Algorithm in Search, Optimization and Machine Learning.* Addison-Wesley, Reading, MA.

Harrington, E. C., Jr. (1965). "The Desirability Function". *Industrial Quality Control* 21, pp. 494–498.

Heredia-Langner, A. (2001). "Genetic Algorithms in Quality Control". Ph.D. dissertation, Arizona State University.

Heredia-Langner, A.; Carlyle W. M.; and Montgomery, D. C. (2003). "Genetic Algorithms for the Construction of D-Optimal Designs". *Journal of Quality Technology* 35(1), pp. 28–46.

Holland, J. (1974). *Adaptation in Natural and Artificial Systems.* The University of Michigan Press, Ann Arbor, MI.

Khuri, A. I. and Conlon, M. (1981). "Simultaneous Optimization of Multiple Responses Represented by Polynomial Regression Functions". *Technometrics* 23, pp. 363–375.

Kros J. F. and Mastrangelo, C. M. (2001). "Comparing Methods for Multi-Response Design Problem". *Quality and Reliability Engineering International* 17, pp. 323–331.

Mayer, D. G.; Belward, J. A.; and Burrage, K. (2001). "Robust Parameter Settings of Evolutionary Algorithms for the Optimization of Agricultural Systems Models". *Agricultural Systems* 69, pp. 199–213.

Montgomery, D. C. (1999). "Discussion of 'Response Surface Methodology-Current Status and Future Directions' by Myers, R. H.". *Journal of Quality Technology* 31, pp. 45–46.

Myers, R. H. and Montgomery, D. C. (2002). *Response Surface Methodology: Process and Product Optimization Using Designed Experiments, 2nd ed.* John Wiley & Sons, Inc., New York, NY.

Nelder, J. A. and Mead, R. (1965). "A Simplex Method for Function Minimization". *Computing Journal* 7, pp. 308–313.

Pignatiello, J. J., Jr. (1993). "Strategies for Robust Multiresponse Quality Engineering". *IIE Tansactions* 25, pp. 5–15.

Reed, P.; Minsher, B.; and Goldberg, D. E. (2000). "Designing a Competent Simple Genetic Algorithm for Search and Optimization". *Water Resources Research* 36(12) pp. 3757–3761.

Reklaitis, G. V.; Ravindran, A.; and Ragsdell, K. M. (1983). *Engineering Optimization, Methods and Applications.* John Wiley & Sons, New York, NY.

Schwefel, H. P. (1997). "Collective Phenomena in Evolutionary Systems". *Preprints of the 31st Annual Meeting of the International Society for General System Research, Budapest,* vol. 2, pp. 1025–1033.

Vining, G. G. (1998). "A Compromise Approach To Multiple Optimization". *Journal of Quality Technology* 30, pp. 309–313.

Wehrens, R.; Pretsch, E.; and Buydens, L. M. C. (1999). "The Quality of Optimization by Genetic Algorithms". *Analytica Chimica Acta* pp. 265–271.