An Improved Genetic Algorithm Using a Directional Search

WEN WAN and JEFFREY B. BIRCH*

Virginia Polytechnic Institute and State University, Blacksburg, VA 24061-0439

The genetic algorithm (GA), a very powerful tool used in optimization, has been applied in various fields including statistics. However, the general GA is usually computationally intensive, often having to perform a large number of evaluations of an objective function. This paper presents four different versions of computationally efficient genetic algorithms by incorporating several different local directional searches into the GA process. These local searches are based on using the method of steepest descent (SD), the Newton-Raphson method (NR), a derivative-free directional search method (denoted by "DFDS"), and a method that combines SD with DFDS. Some benchmark functions, such as a low-dimensional function versus a high-dimensional function, and a relatively bumpy function versus a very bumpy function, are employed to illustrate the improvement of these proposed methods through a Monte Carlo simulation study using a split-plot design. A real problem related to the multi-response optimization problem is also used to illustrate the improvement of these proposed methods over the traditional GA and over the method implemented in the Design-Expert statistical software package. Our results show that the GA can be improved both in accuracy and in computational efficiency in most cases by incorporating a local directional search into the GA process.

KEY WORDS: Genetic Algorithm (GA); Method of Steepest Descent (SD); Monte Carlo (MC) Simulation; Newton-Raphason Method (NR); Split-Plot Design.

Dr. Wan is an assistant professor in Biostatistics and Bioinformatics Unit, Comprehensive Cancer Center, at the University of Alabama at Birmingham. Her email address is wen.wan@ccc.uab.edu.

Dr. Birch* is a corresponding author, a professor in the Department of Statistics at the Virginia Polytechnic Institute and State University. His email address is jbbirch@vt.edu.

Introduction

A genetic algorithm (GA) is a stochastic optimization tool whose search technique is based on the principals of Darwinian survival of the fittest in biological genetics. The GA, originally developed by Holland (1975), simulates an evolutionary process of a living species, using typical biological genetics operations such as "selection", "mutation" and "crossover". GAs have been applied to a broad variety of fields, including ecology, psychology, biochemistry, biology, computational mathematics, and statistics (*e.g.*, Haupt and Haupt, 2004; Heredia-Langner et al., 2003).

The reason that a GA is so popular and useful is that a GA has some attractive features and advantages (Holland, 1992; Haupt and Haupt, 2004), such as employing multiple concurrent search points (not a single point), not requiring the derivative of an objective function, and being able to find a global or near-global optimum of an objective function with a very complex surface and/or in very high-dimensional domains of the function. A disadvantage of the GA, however, is that it is computationally intensive (Haupt and Haupt, 2004). Typically a GA, in order to find the optimum, must evaluate an objective function a large number of times. For example, if taking 12 hours for only a single evaluation of a complex objective function (which is not unusual in applications), then it could be imagined that the GA would become very time-consuming.

To deal with the computational problem, this paper proposes and evaluates four versions of a more computationally efficient GA based on modifying a traditional GA. The main idea of each version of the modified GAs (MGAs) is to gather numerical information from the GA itself so that a local directional search may be used to make computational improvements to the traditional GA. Four local directional searches used in our MGAs include the method of steepest descent (SD), the Newton-Raphson method (NR), a derivative-free directional search method (DFDS), and a method that combines SD with DFDS.

The remainder of this paper is organized as follows. We first briefly review a traditional GA and its operations. The four local directional search methods are discussed next. We then propose our four MGAs. We present some results for several objective functions giving paired comparisons of the GA and the MGAs across a variety of level combinations of the GA operations and two different stopping rules. A real case study where the GA is compared to the MGAs is also illustrated. Finally, we give a summary and conclusions, and suggestions for future work.

The Genetic Algorithm

Genetic algorithms are iterative optimization procedures that repeatedly apply GA operations (such as selection, crossover and mutation) to a group of solutions until some criterion of convergence has been satisfied. In a GA, a search point, a setting in the search space, is coded into a string which is analogous to a *chromosome* in biological systems. The string/chromosome is composed of characters which are analogous to *genes*. In a statistical application, the chromosome corresponds to a particular setting of k factors (or regressors), denoted by $\mathbf{x} = [x_1, x_2, ..., x_k]'$ in the design space and i^{th} gene in the chromosome corresponds to x_i, the value of the i^{th} regressor. A set of multiple concurrent search points or a set of chromosomes (or individuals) is called a *population*. Each iterative step where a new population is obtained is called a *generation*.

A basic GA procedure has the following steps.

1. Define an objective/fitness function, and its variables. Set GA operations (such as population size, parent/offspring ratio, selection method, number of crossovers and mutation rate).

2. Randomly generate initial population.
3. Evaluate each individual (or chromosome) in the initial population by the objective function.
4. Generate an offspring population, by GA operations (such as selection/mating, crossover, and mutation).
5. Evaluate each individual in the offspring population by the objective function.
6. Decide which individuals to include in the next population. This step is referred to as "replacement" in that individuals from the current parent population are "replaced" by a new population, whose individuals come from the offspring and/or parent population.
7. If a stopping criterion is satisfied, then the procedure is halted. Otherwise, go to Step 4.

GAs are a large family of algorithms that have the same basic structure but differ from one another with respect to several strategies such as stopping rules and operations which control the search process. Based on previous experiences, in this study, we use a continuous GA where chromosomes are coded as continuous measurement variables. We also make the following assumptions. The (parent) population size is $2k$ and the offspring population size is also $2k$. The type of selection we utilize is random pairing. The blending crossover is utilized and the number of crossover points depends on the number of dimensions of a specific objective function. Random uniform mutation is utilized and the mutation rate is set around or equal to $1/k$. The type of replacement over both parent and offspring population is ranking or tournament. There are two stopping rules used in this study. Stopping rule 1 is that the GA is halted at the pre-selected number of generations. Stopping rule 2 is that the GA is halted when a cutoff value (which is pre-selected and considered as a near-global value) is achieved. For details on the setting of the GA operations, see, for example, Goldberg (1989), Hamada et al. (2001), Mayer, Belward and Burrage (2001), Francisco Ortiz et al. (2004) and Haupt and Haupt (2004).

Local Directional Search Methods

The GA itself does not utilize a directional search explicitly. In order to improve the computational efficiency of the GA, we modify the GA by incorporating a directional search into the GA process. As mentioned in the introduction, we use four different methods of a local directional search to develop the four MGAs: the method of steepest descent (SD), the Newton-Raphson method (NR), the method of a derivative-free directional search (DFDS), and the method that combines SD and DFDS. The four local search methods will be discussed in the next four subsections, respectively.

The Method of Steepest Descent

The method of the steepest descent (SD) was originally introduced by Cauchy in 1874. It starts at an arbitrary point on the surface of an objective function, $f(\mathbf{x})$, where f is the objective function and $\mathbf{x}$ is the arbitrary point, and minimizes along the direction of the gradient. The simple formula for the $(n+1)^{th}$ iteration at location $\mathbf{x}_n$ (where $\mathbf{x}_n = [x_{n1}, ..., x_{nk}]'$) is given by

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \rho_n \nabla f(\mathbf{x}_n), \tag{1}$$

where ρ_n is a non-negative scalar and $\nabla f(\mathbf{x}_n) = [\partial f/\partial x_{n1}, ..., \partial f/\partial x_{nk}]'$ is the gradient-based vector. Obviously, each step by SD requires the first derivative of f to calculate a specialized gradient based on that particular location. Note that if one wants to find a maximum of f and to maximize along the direction of the gradient, then the ρ_n should be non-positive. More details on SD can be seen in Haupt and Haupt (2004).

The Newton-Raphson Method

Newton-Raphson method (NR), a second directional search procedure, is based on a first-order Taylor series expansion of the function about the point $\mathbf{x}_n$ given by

$$f(\mathbf{x}) \approx f(\mathbf{x}_n) + (\mathbf{x} - \mathbf{x}_n)'\nabla f(\mathbf{x}_n), \tag{2}$$

where $\mathbf{x}$ is some point near $\mathbf{x}_n$. To find an optimal value of f, taking the gradient of both sides of (2) and setting it equal to zero yields

$$\nabla f(\mathbf{x}) \approx \nabla f(\mathbf{x}_n) + \mathbf{H}_n(\mathbf{x} - \mathbf{x}_n) \equiv \mathbf{0},$$

where $\mathbf{H}_n$ is the Hessian matrix with elements given by $h_{njl} = \partial^2 f/\partial x_{nj}\partial x_{nl}$, j and $l = 1, ..., k$. Thus the next point, $\mathbf{x}_{n+1}$, can be found by

$$\mathbf{x}_{n+1} \approx \mathbf{x}_n - \mathbf{H}_n^{-1}\nabla f(\mathbf{x}_n). \tag{3}$$

More details on NR can be seen in Haupt and Haupt (2004).

Compared to SD in (1), NR requires calculating the Hessian matrix (which involves the second derivative of f) and its inverse and thus it usually takes more time than SD for each function evaluation. However, NR does not require the adjustment to the moving step (ρ_n in formula (1)) as SD does, since $-\mathbf{H}_n^{-1}$ takes the amount of the moving step into account. In practice, the NR method often requires fewer steps than the SD method to converge to an optimal solution.

A Derivative-free Directional Search Method

The SD and NR methods both require the partial derivatives of an objective function f. It is not expected that SD or NR can always find a proper direction from the current point, since an objective function usually is not simple and unimodal, but very complicated, locally rough and unsmoothed. Thus, we developed a new local directional search method which is derivative-free and denoted by "DFDS."

The goal of DFDS is to find an appropriate direction so as to build the path without requiring the gradient, $\nabla f(\mathbf{x}_n)$. Here we build three potential directions associated with the best offspring in a GA process. When the best offspring is also the best in the current parent population, there is an improvement from its parents to the best offspring in terms of the objective function. It may be possible to make continuous improvements by moving along the directions/paths from its parents to the best offspring. That is, some data points are "collected" along the paths until no further improvement can be found.

When the best offspring among both the offspring and parent populations is found, we can trace back to find its parents. These parents then can be considered as two different starting points. Both of their first steps from the two starting points go to the same point: the best offspring. So two directions are established: one is from one of the parents to the best offspring; the other is from the second of the parents to the offspring. Both directions have obtained improvement, since the best offspring of interest is an improvement over both its parents in terms of values of an objective function.

For example, consider a 2-dimensional ($k = 2$) problem along with the contours of a response (or values of an objective function) as illustrated in Figure 1. In general, the best offspring among the offspring and the current parent population is denoted by O (expressed as $\mathbf{x}_O = [x_{O1}, ..., x_{Ok}]'$) and its parents are denoted by P1 ($\mathbf{x}_{P1} = [x_{P11}, ..., x_{P1k}]'$) and P2 ($\mathbf{x}_{P2} = [x_{P21}, ..., x_{P2k}]'$). Obviously, there are two directions: one is from P1 to O, expressed as $\delta_{P1O} = \mathbf{x}_O - \mathbf{x}_{P1} = [\delta_{11}, \delta_{12}, ..., \delta_{1k}]'$ and the other is from P2 to O, expressed as $\delta_{P2O} = \mathbf{x}_O - \mathbf{x}_{P2} = [\delta_{21}, \delta_{22}, ..., \delta_{2k}]'$. We refer to these two directions as the Parent 1 and Parent 2 directions.

The third direction is the "common" direction, expressed as $\delta = [\delta_{31}, \delta_{32}, ..., \delta_{3k}]'$, and based on the two parent directions. If δ_{1i} and δ_{2i}, for $i = 1, ..., k$, are both positive (negative), then δ_{3i} is positive (negative). That is, if both the parent directions are in common, say, both positive (negative) along the X_i axis, then the third direction is positive (negative) along the X_i axis. If δ_{1i} and δ_{2i} are opposite in direction, then δ_{3i} is set to 0. That is, if the parent directions are not in common on the X_i axis, then the third direction has no movement along the X_i axis. For more details on the three directions and determining their moving distances for each moving step, see Appendix A.

Figure 1 illustrates the three defined directions. The optimal point is denoted by "Θ". It is easy to see the two parents directions, expressed as $\delta_{P1O} = [\delta_{11}, \delta_{12}]'$ and $\delta_{P2O} = [\delta_{21}, \delta_{22}]'$ respectively. The third direction $\delta = [\delta_{31}, \delta_{32}]'$. Obviously, $\delta_{31} > 0$ since both $\delta_{11} > 0$ and $\delta_{21} > 0$. That is, the common direction in this case is positive along the X_1 axis. And $\delta_{32} = 0$ since $\delta_{12} > 0$ and $\delta_{22} < 0$. That is, the common direction has no relative movement along the the X_2 axis.

[Insert Figure 1 about here.]

Once the three directions are defined, starting at O, the DFDS method moves along the three directions/paths, with some appropriate moving distance for each moving step until no improvement is found in terms of an objective function. In Figure 1, the three "stars" on the paths denote that the three best points found on each path and the processes of moving along the paths will be stopped at their next points due to no further improvement.

A Method Based on Combining SD and DFDS

Unlike the SD, NR, DFDS methods, the fourth method we used in this study is a "combined" method that combines SD, a derivative-base search method with one direction generated, with, DFDS, a derivative-free search method with three directions generated. Therefore, this method provides a total of four directions to search for the best point.

A Summary of the Methods of a Local Directional Search

In summary, the four local directional search methods used in our four MGAs in this study are SD, NR, DFDS, and the method that combines SD with DFDS. There are many other MGAs that may be considered by using other derivative-based directional searches combined with other derivative-free directions.

We choose these four local directional search methods for our four MGAs because we have the following concerns: (1) SD is quite simple, efficient, but requires the first derivative of f. (2) NR is a very popular optimization tool but requires calculating the Hessian matrix and its inverse matrix. Thus, it may take much more time than SD for each function evaluation. (3) DFDS with the three directions generated is intuitive, reasonable, and derivative-free. (4) For the method that combines SD with DFDS, we want to determine if such a combination performs better than either the SD or the DFDS, separately.

Modified Genetic Algorithms

We developed four versions of a modified genetic algorithm (MGA). These MGAs are listed as follows: (1) if a directional search by SD is utilized by the GA process, then the MGA is denoted by "MGA_{SD};" (2) if a directional search by NR is utilized, then the MGA is denoted by "MGA_{NR};" (3) if a directional search with the three directions, the DFDS method, is utilized, then the MGA is denoted by "MGA_3;" (4) if a directional search with a total of four directions combined by SD and DFDS, then the MGA is denoted by "MGA_4."

These MGAs have the same main idea: utilizing numerical information from a GA process itself to find some appropriate local directions by only requiring a few extra function evaluations so that the GA process may be guided to further possible improvement. The numerical information we utilized in our study is focused on the best offspring among both the current parent and offspring populations.

The general procedure for each MGA is the same as that of GA, except that in the i^{th} generation we add Step D between Step 5 and 6 in the original GA procedure as follows:

- D. Is the best offspring in the offspring population also the best over the current parent population?
    - D-1. If no, directly go to Step 6.
    - D-2. If yes, then define and implement a local direction. Collect data points along the paths with some appropriate moving distance until no improvement is observed in the objective function. Find the best point and replace the best offspring by the best point. Then go to Step 6.

The choice of the size of an appropriate moving distance, d, depends on how bumpy the surface of an objective function is. If the surface is very bumpy relative to the region of the domain, then the appropriate d should be relatively small. Otherwise, the appropriate d should be relatively large to make the MGAs more efficient.

Actually, the general MGA process is a special GA process with an extra “branch” (illustrated in Step D) (i.e. requiring only a few extra function evaluations), where the best offspring which is also the best over the current parent population is found. Within the branch, a local direction can be defined by SD, NR, DFDS, or the method that combines SD and DFDS. Along the direction(s)/path(s), data points are collected (i.e. evaluated in terms of an objective function) with an appropriate moving distance for each moving step until no further improvement is found. The best offspring from the parent population is replaced by the best point found on the paths. Then the branch is ended with possible improvement for the MGA by replacing the best offspring with the new best point found and the MGA process is continued like a GA process until a new extra “branch” is found and generated. That is, a new best offspring, which is also the best in a new current parent population, is found. The whole process is iterated until some appropriate stopping rule is satisfied.

Each MGA is essentially a modification to a GA. Thus, if the GA can jump out of a local optimum, so can the MGAs. In addition, each MGA will more likely produce an improved solution than that obtained by the GA with the same setting of the GA operations. An improved solution results when, under the same situation and the same stopping rule, the best solution found by each MGA is closer to the true global solution (in accuracy) and/or converges faster to a global optimum than by the GA (in computational efficiency).

Computational details for implementation of a directional search into a GA process by the SD, NR, and DFDS methods are found in Appendixes B, C and A, respectively. Details for implementation by the method that combines SD and DFDS are straightforward from the details of implementation by SD and DFDS.

Examples

In our examples, the main goal is to compare the four MGAs with the GA in computational efficiency and in accuracy for different objective functions under a variety of combinations using different levels of GA operations. At the same time, our sub-goal is to find optimal levels for each operation among a variety of levels of interest for each MGA and the GA.

To make the comparisons more fairly comparable, whenever possible, the same random numbers generated within the GA are also used within each version of the MGAs.

Therefore, an experiment is conducted through a split-plot design (Hinkelmann and Kempthorne, 1994) so that paired comparisons can be made under the same settings of the operations and using the same random numbers.

The three whole-plot factors are the three main GA operations: replacement type (denoted by "type" in subsequent references), crossover points (denoted by "crossover"), and mutation rates (denoted by "mutation"). The factor type has two levels: ranking (0) and tournament (1). The factor crossover and the factor mutation have two or three levels, depending on the number of variables used in the objective function. Essentially, these combinations of the three factors correspond to the settings of the three main GA operations. The sub-plot factor is "method" which has five levels: one is the GA (denoted by method = 0) and the other four are MGA$_{SD}$, MGA$_{NR}$, MGA$_3$, and MGA$_4$ (denoted by method = 1, 2, 3, and 4, respectively).

Under the same setting of the GA operations, the GA and the four MGAs may obtain a different optimum value for different random seeds. Therefore, a Monte Carlo experiment is performed for each specific combination of levels of these three GA operations and repeated 500 times using different random seeds.

Two Stopping Rules

Two different stopping rules are utilized in the experiment. Under rule 1, the algorithm will be halted at a pre-selected number of generations. Thus, this stopping rule can be used to compare the five algorithms for accuracy in finding the optimal value of the objective function. Our MGAs all require extra evaluations of the objective function, f, and MGA$_4$ usually requires the most evaluations of f among the four MGAs found in our studies. Thus, under the same random seed and the same settings of the GA operations, we let the traditional GA run the number of evaluations equal to the number of the extra evaluations of MGA$_4$ added onto the pre-selected number of evaluations (which is equal to the number of generations times the population size).

Under rule 2, the algorithm will be halted when the cut-off value, which is close to the optimal value and beyond all-possible local optima, is achieved. In our examples except for the case study, the optimal values are known. Thus, the second rule can be used to compare the five algorithms for efficiency in finding the optimal value of the objective function.

Comparison Criteria

There are three responses of interest used for comparing the four MGAs to the GA. The first one is the best optimal value of an objective function obtained by the GA or MGAs. We denote this optimal value by "best." The second response of interest is the distance from the location of the best value obtained to the location of the true optimal value, denoted by "distance." The third response is the total number of evaluations of the objective function, denoted by "evaluation." Under stopping rule 1, the interesting responses are best and distance. Under stopping rule 2, the most interesting response is evaluation, with best and distance also of interest.

Boxplots (from Minitab) will be our graphical tool to compare the four MGAs to the GA across all the combinations. The numerical criteria utilized for comparison are (1) the mean squared error (MSE) of the responses best and distance, denoted by "MSE(best)" and "MSE(distance)," respectively; (2) the mean and variance of the number of evaluations (the response evaluation), denoted by "Mean(evaluation)" and "Var(evaluation)," respectively; and (3) the number of winners among the 500 replications between any two of the four algorithms for each setting of the operations (or each combination) in terms of the responses best, distance or evaluation, denoted by "Count(best)," "Count(distance)," and "Count(evaluation)," respectively.

For Criteria (1), the MSE is given by

$$MSE = \frac{\sum_{i=1}^{500} (y_i - T)^2}{500}, \tag{4}$$

where y_i is a response (either best or distance) and T is an true optimum for the response y_i. For Criteria (2), the MSE cannot be used for the response evaluation, since no optimum exist for this response. Thus the estimated mean and variance are used as criteria for evaluation in our study. For Criteria (3), among the five algorithms there are a total of ten paired comparisons in terms of the number of winners among the 500 replications for each combination, where a "winner" refers to the most favorable response among each pair of responses being compared. For each paired comparison, there may be some ties when the values obtained by one algorithm are equal to the values by another algorithm. For example, to compare GA versus MGA$_{SD}$ in terms of Count(evaluation), it follows that "Count(evaluation) by GA" + "Count(evaluation) by MGA$_{SD}$" + "Ties(evaluation)" = 500. In the following examples, the numbers of counting ties will not be presented.

Computational time is used to compare the computational efficiency. Besides MGA$_{NR}$, the computational time of a single function evaluation with a local directional search implemented for each of the other three MGAs is not very different from that by GA in our C++ code, especially for the cases with a single function evaluation, a time-consuming task. Thus, the number of evaluations under stopping rule 2 will be an appropriate indirect measurement of total computational time for each of the GA, MGA$_{SD}$, MGA$_3$, and MGA$_4$ procedures.

Comparisons for the Benchmark Functions

For the comparisons of GA, MGA$_{SD}$, MGA$_3$, MGA$_4$, and MGA$_{NR}$, we have selected five objective functions used in previous GA literature. The five objective functions are (1) the sphere model in 2-dimension with smooth surface (Back, 1996), (2) and (3) the Schwefel's function with relatively a bumpy surface (which has been utilized as a benchmark function by Araujo and Assis (2001)) in 5- and 20-dimension respectively, (4) and (5) the Rastrigin's function with a very bumpy surface (another benchmark function by Araujo and Assis (2001)) in 5- and 20-dimension respectively. The results from all these five objective functions show that MGA$_{SD}$, MGA$_3$, MGA$_4$, and MGA$_{NR}$ all perform better in both accuracy and computational efficiency than GA over nearly all combinations and all criteria. The exception is for function (4) (the Rastrigin's function in 5-dimension) in terms of the MSE of the response distance where all MGAs outperform GA in seven combinations out of 18. As an example, we only present some results for function (5) (Rastrigin's function in 20-dimension). This function presents a serious challenge to the GA and the MGAs, due to a very bumpy surface of the function in high dimensions. For details about the sphere model and the Schwefel's function, see Appendix D.

Comparisons for the Rastrigin's function with 20 dimensions

A generalized Rastrigin's function is given by

$$f(\mathbf{x}) = \sum_{i=1}^{k} (x_i^2 - 10\cos(2\pi x_i) + 10), \text{ where } -5.12 \le x_i \le 5.12, \tag{5}$$

where k is the number of dimensions of the function. Figure 2 shows its 1- and 2-dimensional surfaces. The surfaces are very bumpy in a narrow range [-5.12, 5.12]. The goal is to find a minimal value and its corresponding location by GA and MGAs. The minimum of this function is known as $\min(f(\mathbf{x})) = f(0, ..., 0) = 0$. In this study, we compare the five algorithms using the function in 20 dimensions (that is, $k = 20$).

[Insert Figure 2 about here.]

To conduct a split-plot design, the levels of the three whole-plot factors are as follows. The factor type has 2 levels: ranking and tournament; the factor crossover has levels: 2, 4, and 8; and the factor mutation has levels: 0.04, 0.05, and 0.06. There are a total of 18 combinations of type, crossover and mutation. Note the middle level for mutation of 0.05 is $1/k$ where k is the number of genes (or dimensions). For stopping rule 2, the cut-off value, which is a near-global optimum, is set to 0.5. The pre-selected number of generations used by stopping rule 1 is 5,000. The appropriate moving distance for each moving step, d, is set to 0.05.

Under stopping rule 1, Figure 3 presents boxplots for the responses best and distance across the 500 repetitions for the MGAs and GA models for each of the 18 combinations of type, crossover and mutation. This figure illustrates that MGA_{SD}, MGA_3, MGA_4, and MGA_{NR} all perform better than GA over all 18 combinations in terms of the best value and the distance. Not only are all these four new methods more accurate (plots closer to the true minimal value 0), but also more precise (plots exhibit less spread) over all situations. Among the four MGAs, MGA_{NR} performs the best in both accuracy and precision, since the 500 best values obtained by MGA_{NR} all achieve zero (the true minimum) across all of 18 combinations. In addition, MGA_{SD} and MGA_4 both perform much better than MGA_3: when type is 0 (ranking), the best values found by both MGA_{SD} and MGA_4 are all zero across all of the nine combinations shown in the top left boxplot; when type is 1 (tournament), most of the best values by both MGA_{SD} and MGA_4 are zero except for a few outliers shown in the top right boxplot. The boxplots by the response best and by the response distance express similar patterns. That is, lower best values have smaller distances. The numerical results including the MSE of best and distance (which are not presented here) also match well with Figure 3.

[Insert Figure 3 about here.]

Under stopping rule 1, the amounts of time recorded to complete the 500 repetitions for GA, MGA$_{SD}$, MGA$_3$, MGA$_4$, and MGA$_{NR}$ are 22959, 23614, 23120, 23628, and 38616 seconds, respectively. Except for MGA$_{NR}$, the times of the other four algorithms are relatively similar to each other. The slight differences in the amounts of time between the other four are due to the slightly different computations required for each MGA/GA and to the slightly different numbers of extra function evaluations for MGA$_{SD}$, MGA$_3$, and MGA$_4$. The reason that MGA$_{NR}$ took much longer than the other four is in calculating the Hessian and its inverse matrix in formula (3).

Under stopping rule 2, Table 1 presents the mean of the number of function evaluations and its estimated Monte Carlo (MC) error as a summary of the 500 repetitions for comparisons of the five algorithms. It shows that the numbers of evaluations required to obtain a value of the objective function within 0.5 of the true minimum by MGA$_{SD}$, MGA$_3$, MGA$_4$, and MGA$_{NR}$ are all consistently less than required by GA over all combinations. Among the four MGAs, MGA$_{NR}$ performs the best with much smaller mean values for the number of function evaluations than the other three MGAs over all combinations. Among the other three MGAs, MGA$_{SD}$ has the smallest mean values of the number of function evaluations in 12 combinations out of 18, MGA$_4$ has the smallest mean values in five combinations, while MGA$_3$ has the smallest value in only one combination (which is the 17th).

[Insert Table 1 about here.]

Also under stopping rule 2, Table 2 presents the paired comparisons of GA, MGA$_{SD}$, MGA$_3$, MGA$_4$, and MGA$_{NR}$ (denoted by “0, SD, 3, 4, and NR,” respectively) in terms of the number of winners among the 500 replications for each combination with respect to the response evaluation (denoted by "Count(evaluation)"). Note that Table 2 presents only six paired comparisons, not ten (the total number of paired comparisons), because these sixed paired comparisons are sufficient to rank these five algorithms. These paired comparisons show that all MGAs have more winners than GA over all combinations in terms of the count of the number of evaluations. Among the four MGAs, MGA$_{NR}$ has consistently the most winners over all combinations. MGA$_{SD}$ has the most winners than the other two MGAs across all the combinations, and MGA$_4$ has more winners than MGA$_3$ over all combinations.

[Insert Table 2 about here.]

The amounts of time recorded for GA, MGA$_{SD}$, MGA$_3$, MGA$_4$, and MGA$_{NR}$, under stopping rule 2, are 5622, 3717, 4137, 3702, and 15 seconds, respectively. Obviously, MGA$_{NR}$ finds the optimal solution very quickly, with MGA$_{SD}$ and MGA$_4$ as the next fastest, while the GA is the slowest. These results match well with those in Tables 1 and 2.

Some Other Details on Comparisons Among the Four MGAs using the Benchmark Functions

Among our four MGAs, in the examples of the benchmark functions, MGA$_{NR}$ performs the best in terms of our criteria except for the amount of time recorded under stopping rule 1 for the Rastrigin's function with very bumpy surface in 5- and 20-dimensions. Among the other three MGAs, under stopping rule 1, MGA$_{SD}$ and MGA$_4$ are quite competitive with each other and both perform much better than MGA$_3$ in most cases. In addition, under stopping rule 2, MGA$_{SD}$ performs better than MGA$_4$ and MGA$_3$, and MGA$_4$ performs better than MGA$_3$ in most cases.

In the examples using Rastrigin's function in 5- or 20-dimensions, the results show that MGA$_{NR}$ exhibits superior performance over the other three MGAs, especially when using stopping rule 2. When under stopping rule 1, MGA$_{NR}$ took much more time to finish the MGA process than the other three MGAs, although MGA$_{NR}$ still has the best performance by far in terms of other criteria such as MSE(best) and MSE(distance). Except for the time concern, it seems that the local directional search using NR greatly helps the GA process jump out of local peaks or valleys towards the global optimum. But this superior performance by MGA$_{NR}$ appears to hold only for a function with a very bumpy surface. When the Schwefel's function is used with its relatively bumpy surface, the results (not presented in this paper) show MGA$_{NR}$ still performs better than MGA$_{SD}$, but both algorithms are very competitive with each other in terms of all criteria including the amount of time taken. It seems that when the peaks or valleys are further away from each other, the search by NR does not easily jump over them as when the peaks or valleys are quite close to each other.

Comparisons for the Case Study: A Chemical Process

The real example used to illustrate our methods is taken from Myers and Montgomery (2002), where a central composite design (CCD) was conducted on a chemical process. Two independent variables (or factors) are time (x_1) and temperature (x_2). Three responses of interest are yield (y_1), viscosity (y_2) and number-average molecule weight (y_3). The collected data are given in Myers and Montgomery (2002). As in Myers and Montgomery (2002), we transform the natural independent variables into the coded variables within the range of [0, 1].

In this case study, their multi-response optimization goal is to maximize y_1 (the minimum L = 70 and optimum T = 80), achieve a target value for y_2 (the minimum

L = 62, the target T = 65, and the maximum U = 68), and, at the same time, control y_3 within the acceptable range of [3200, 3400]. The desirability function method by Derringer and Suich (1980) is utilized to find simultaneous optimum solutions of the responses y_1, y_2, and y_3.

The desirability function (which is the objective function utilized in GA and the MGAs) is given by

$$D = (d_1 \times d_2 \times \cdots \times d_m)^{1/m} \times 100\%, \tag{6}$$

where m is the number of responses and d_i is the i^{th} individual desirability, which is given in Derringer and Suich (1980). The researcher's goal is to find the common location, $\mathbf{x}$, where the maximum value of D is achieved, indicating, in some way, the best location, $\mathbf{x}$, where all the responses achieved their most desirable values simultaneously. In addition, the solution vector, $\mathbf{x}_s$, should be controlled within the experimental region R, which is defined as $(x_1 - 0.5)^2 + (x_2 - 0.5)^2 \leq 0.5^2$ in this case study.

Under the same conditions such as experimental priority and fitted models given in Myers and Montgomery (2002), the two solutions we found by GA are listed as follows.

1) $x_1 = 0.5758$ $x_2 = 0.1624$ $\hat{y}_1 = 78.6344$ $\hat{y}_2 = 65.0000$ $\hat{y}_3 = 3260.7992$ $D = 0.9292$
2) $x_1 = 0.2661$ $x_2 = 0.7964$ $\hat{y}_1 = 78.2694$ $\hat{y}_2 = 65.0000$ $\hat{y}_3 = 3399.1632$ $D = 0.9094$

These two solutions are different from the two solutions obtained by Design-Expert as shown in Myers and Montgomery (2002) (whose two values of D are 0.822 and 0.792) in terms of fitted optimal values for all of the three responses. The solutions obtained by GA result in larger values of D, indicating that GA performs better at finding the optimal value of D than the algorithm used by Design-Expert in this example.

Figure 4 represents the surface (the left graph) of the desirability function D within the experimental region R and its corresponding contour plot (the right graph). The figure shows that there are two distinct surfaces which represent two disjoint operating regions. Obviously, the surface of D matches well to the contour plot. In addition, the two optimal solutions we found also match well to the figure. Notice that if the case study had more than two or three factors/dimensions, then it would be hard to graphicly show the surface of the desirability function D and its contour plot. Thus, in such a situation, we could not depict graphically the location of the optimal solution. But we still could use either the MGAs or GA to find its optimal or near-optimal solution.

[**Insert Figure 4 about here.**]

To compare the performance of GA, MGA$_{SD}$, MGA$_3$, MGA$_4$, and MGA$_{NR}$ on this example, a split-plot design is conducted and repeated 500 times, similar to the design used in the five examples mentioned above. The pre-selected number of generations was set at 50. The appropriate moving distance was set at 0.001. Since the true optimal solution is unknown, the response "distance" cannot be measured in this application. Stopping rule 2 is also not suitable in this example, since the pre-specified cutoff which is a near-global optimal value is unknown. We consider only two levels of the factor crossover instead of three considered in the previous example. Thus, there are only 12 combinations of our three factors, type, crossover, and mutation. To calculate the MSE of the response "best" of the desirability function D, MSE(best), for each combination with 500 repetitions, based on formula (2), we need the value of T, the true optimum (the maximum of D), which is, however, unknown in this case. Since the maximum of D is generally close or equal to one, T is set to be one for this example.

Table 3 presents the results with respect to the MSEs of the response "best" of the desirability function D and the estimated MC error for each combination under stopping rule 1 for this case study. It shows that MGA$_4$ has the smallest MSEs among the five algorithms over all 12 combinations. MGA$_3$ has the next smallest MSEs over all combinations. MGA$_{SD}$ has smaller MSEs than GA in six combinations, while MGA$_{NR}$ has only one smaller MSE value than GA.

[Insert Table 3 about here.]

Also under stopping rule 1, Table 4 presents the results on the six paired comparisons of GA, MGA$_{SD}$, MGA$_3$, MGA$_4$, and MGA$_{NR}$ in terms of the number of winners among the 500 replications for each combination with respect to the response best (denoted by "Count(best)"). For the same reason as Table 2, this table presents only six paired comparisons, not ten. These paired combinations show that MGA$_4$ has superior performance since it consistently has more winners than the other four over all combinations. MGA$_3$ performs the second best since it has more winners than GA, MGA$_{SD}$, and MGA$_{NR}$ over all combinations. MGA$_{SD}$ is the third best since it has more winners than GA and MGA$_{NR}$ in most combinations. However, MGA$_{NR}$ is even worse than GA over all combinations.

[Insert Table 4 about here.]

The results in Table 4 match well with those in Table 3. The amounts of times recorded for GA, MGA$_{SD}$, MGA$_3$, MGA$_4$, and MGA$_{NR}$ are 54, 47, 52, 53, and 49 seconds, respectively.

Unlike the results for the benchmark functions, MGA$_{SD}$ and MGA$_{NR}$ both perform worse than MGA$_3$. We speculate that one reason for this result is that the surface of the desirability function in the case study has two disjoint "mountains", both of which are locally irregular, unlike the surfaces of the five objective functions which are locally smooth and regular. One reason that MGA$_{NR}$ is worse than GA under stopping rule

1 is that the number of function evaluations required by MGA$_{NR}$ is less than the total of number of evaluations run by GA, which is equal to the number of extra evaluations of MGA$_4$ added onto the pre-selected number of evaluations. When we let GA run the number of evaluations equal to the number of the extra evaluations of MGA$_{NR}$ (which is smaller than the number of the extra evaluations of MGA$_4$) added onto the pre-selected number of evaluations, the results show that MGA$_{NR}$ has the smaller MSEs than GA in eight combinations out of 12.

Summary on the GA/MGAs Optimal Settings from the Examples

Recall that our goal for this study is to find optimal levels for each operation among a variety of levels of interest to the user of either MGAs or GA. A Monte Carlo experiment has been performed for each combination of levels of the three GA operations (type, crossover and mutation). In this study, the optimal settings for each algorithm are decided based on the MSEs of the response “best” when using stopping rule 1, and based on the mean of the response “evaluation” when using stopping rule 2.

Table 5 presents the summary of the optimal settings for GA, MGA$_{SD}$, MGA$_3$, MGA$_4$, and MGA$_{NR}$ from the examples including the case study. The first row of this table says that in the case study with two factors, under stopping rule 1, the optimal setting for all of the five algorithms is tournament replacement, one crossover point, and 0.6 mutation rate. The presentations of the other rows follow the format of the first row.

[Insert Table 5 about here.]

From Table 5, it seems that under the different stopping rules, each specific example has its own optimal GA setting for each of the five algorithms. These results seem to agree with the "No Free Lunch Theorems for Optimization" conclusions by Wolpert and Macready (1997), which states that the optimal GA setting is problem-dependent and there are no general optimal GA settings.

However, from Table 5, there are some rules we may follow before either the MGAs or GA are run. First, the factors crossover and mutation both depend on the length of a chromosome/string (which is the number of genes in a chromosome). Second, ranking, a replacement type, is preferred in most cases, especially when the surface of an objective function is bumpy or very bumpy. Third, the factor crossover is important and the number of crossover points should be increased as the length of a chromosome increases. Fourth, the optimal mutation rate is approximately equal to $1/k$, as suggested in Back (1996).

Conclusion and Discussion

This paper presents the four versions of modified GAs: MGA$_{SD}$, MGA$_3$, MGA$_4$, and MGA$_{NR}$, all of which make an improvement over the traditional GA both in accuracy (by stopping rule 1) and in computational efficiency (by stopping rule 2) in most cases. The main idea in our modification is to implement a local directional search into the GA process. The local directional searches utilized in this paper to develop our four MGAs include using SD, NR, DFDS, and the method that combines SD with DFDS. MGA$_{SD}$ and MGA$_4$ both require the first derivative of f, MGA$_{NR}$ requires calculating the Hessian matrix with the second derivative of f and its inverse matrix, while MGA$_3$ requires no derivative calculations.

Several examples, including a case study of a chemical process, are used to facilitate comparisons of GA, MGA$_{SD}$, MGA$_3$, MGA$_4$, and MGA$_{NR}$. Such examples include comparisons between low-dimensional and high-dimensional problems, and smooth, relatively bumpy and very bumpy surfaces. Numerical and graphic comparison results in all of the examples show that the new MGAs procedures perform better than the traditional GA procedure in most cases.

Among the four MGAs, the results show that MGA$_{NR}$ performs the best in the examples using the benchmark functions (Araujo and Assis, 2000) in terms of all comparison criteria, except for the amount of time taken under stopping rule 1 for the benchmark function with a very bumpy surface. Under stopping rule 2, MGA$_{NR}$ demonstrates a considerable improvement over the other MGAs regarding all criteria including the amount of time when using the benchmark function with a very bumpy surface. However, when using the benchmark functions with relatively a less bumpy surface (like the Schwefel's function) or a smooth surface (like the spherical model), MGA$_{NR}$ and MGA$_{SD}$ are quite competitive in terms of all criteria including the amount of time.

Among the other three MGAs, under stopping rule 1, the comparison results in the examples of the benchmark functions show that MGA$_{SD}$ and MGA$_4$ are competitive with each other and both perform much better than the MGA$_3$ in most situations. Under stopping rule 2, the comparison results show that MGA$_{SD}$ performs the best with MGA$_4$ performing better than MGA$_3$ in most situations. In summary, for our benchmark functions, MGA$_{NR}$ is the top method, followed in order by MGA$_{SD}$, MGA$_4$, and MGA$_3$.

However, the results in the case study are quite different from those in the examples based on the benchmark functions. In the case study, the results show that in terms of all criteria, MGA$_4$ exhibits superior performance, followed in order by MGA$_3$, MGA$_{SD}$ and MGA$_{NR}$. We speculate that one reason that MGA$_{SD}$ and MGA$_{NR}$ both perform worse than MGA$_3$ is that the surface of the function in the case study has two disjoint "mountains", both of which are locally irregular, unlike the surfaces of the benchmark functions which are locally smooth and regular.

Based on all the results in the examples including the case study, we prefer to use MGA$_4$ if the first derivative can be taken for an objective function f. If the second derivative can be taken for f and if the surface of f is very bumpy but locally smooth and regular, then we would choose MGA$_{NR}$. But if derivative cannot be taken for f, then MGA$_3$ is the only suitable choice.

Several issues remain for further study. For example, the three derivative-free directions defined in MGA$_3$ may not be optimal. Additionally, the derivative-based directions defined in MGA$_{SD}$ and MGA$_{NR}$ may also not be optimal. Perhaps, there are other directions better than the four we have chosen in this paper. Another issue concerns the appropriate moving distance, once the directions are chosen. The size of an appropriate moving distance, arbitrarily chosen by us, may greatly affect the efficiency of the MGAs. The last issue is on the optimal setting of the GA operations. In this study, type of replacement, the number of crossover points, the mutation rate, the three main GA operations, have been studied. However, there may be some other operations affecting the GA performance, such as population size and parent/offspring ratio. We plan to study these issues in future work.

C++ code is available upon request from the authors.

APPENDIX A

Mathematical Representation of the Three Directions in MGA$_3$

We first introduce our notation. Parent 1 (P1) is given by $\mathbf{x}_{P1} = [x_{P11}, ..., x_{P1k}]'$, where $\mathbf{x}$ is a vector of size $k \times 1$ where k is the number of factors or the number of dimensions. Similarly, Parent 2 (P2) is given by $\mathbf{x}_{P2} = [x_{P21}, ..., x_{P2k}]'$, and their offspring (O) is expressed as $\mathbf{x}_O = [x_{O1}, ..., x_{Ok}]'$. The Parent 1 direction (from P1 to O) is expressed as δ_{P1O} and the Parent 2 direction (from P2 to O) is as δ_{P2O}. And the common direction is simply denoted as δ. The new points after the first step along the three directions are expressed as $\mathbf{x}_{New1} = [x_{New11}, ..., x_{New1k}]'$, $\mathbf{x}_{New2} = [x_{New21}, ..., x_{New2k}]'$, and $\mathbf{x}_{New} = [x_{New1}, ..., x_{Newk}]'$, corresponding to Parent 1, Parent 2, and their common direction respectively. The appropriate moving distance on each axis in each moving step is expressed as d.

The parent 1 direction, which essentially is the different distances on each dimension between points P1 and O, is expressed as

$$\delta_{P1O} = \mathbf{x}_O - \mathbf{x}_{P1} = [\delta_{11}, \delta_{12}, ..., \delta_{1k}]'. \tag{7}$$

Similarly, the parent 2 direction is expressed as

$$\delta_{P2O} = \mathbf{x}_O - \mathbf{x}_{P2} = [\delta_{21}, \delta_{22}, ..., \delta_{2k}]'. \tag{8}$$

To keep the same directions and move along the three paths, the moving distance on each axis should be in constant proportion to each other, as the method of steepest ascent/descent in response surface methodology (RSM). [In RSM, the constant proportion on the i^{th} dimension is defined as $\hat{\beta}_i/\hat{\beta}^*$, where the $\hat{\beta}_i$ is the ith estimated coefficient in the estimated first-order model and the $\hat{\beta}^*$ is the largest coefficient in magnitude among the k estimated coefficients, that is, $\hat{\beta}^* = \max\limits_{i=1,...,k} |\hat{\beta}_i|$. From this ratio, we can see that the proportion only depends on the β_i, the ith coefficient. The moving distance on the ith dimension is defined as $(\hat{\beta}_i/\hat{\beta}^*) * \rho$, where the ρ is an appropriate fixed distance. (For more details, please see Myers and Montgomery (2002) in page 205-207.)]

In our GA application, the main idea in moving along the parent 1 path is the same as that in the method of steepest ascent/descent. That is, to keep the constant proportion in each dimension and move some appropriate fixed distance (which is d in our case) along the parent 1 path. But the difference between our GA case and RSM is the starting point. In the GA case, the starting point is P1, not O. That is, the first step has already been completed. So the next moving step starts at O. The largest moving distance in the first step is also not d, but $\max_{i=1,\ldots,k} |\delta_{1i}|$, where the δ_{1i} is the moving distance on ith axis in Equation (A.1). Let δ_1^* denote $\max_{i=1,\ldots,k} |\delta_{1i}|$. In our study, if $\delta_1^* < d$, then the moving distant in the next step will be δ_1^*. Otherwise, the distance in the next step will be d. The distance d is obviously utilized to control the next moving distance.

The procedure of moving along the parent 1 direction is as following.

1. Calculate δ_{P1O} and then find $\delta_1^* = \max_{i=1,\ldots,k} |\delta_{1i}|$, the largest distance in the first moving step.

2. If $\delta_1^* < d$, then the next new position on the ith axis, $i = 1, ..., k$, is defined as $x_{New1i} = x_{Oi} + (\delta_{1i}/\delta_1^*) \times d$. Otherwise, the new position is $x_{New1i} = x_{Oi} + \delta_{1i}$.

3. Check the region of the new point $\mathbf{x}_{New1} = [x_{New11}, ..., x_{New1k}]'$. If x_{New1i} is greater than its upper bound (which is the largest value in the ith domain), then let it be the upper bound . Similarly, if it is less than its lower bound (which is the lowest value in the ith domain), then let it be the lower bound. (Usually, the upper bounds and lower bounds have been given through defining the objective function.)

4. Evaluate the new point $\mathbf{x}_{New1}$ by the objective function. If the new point performs worse than the point $\mathbf{x}_O$, then the process of moving along the parent 1 direction is halted. If the new point performs better than the $\mathbf{x}_O$, then replace the point $\mathbf{x}_{New1}$ by the next new point $\mathbf{x}_{New1} + \mathbf{\Delta}_{N1O}$, where $\mathbf{\Delta}_{N1O} = \mathbf{x}_{New1} - \mathbf{x}_O$. (The "N1O" means "New point from Parent 1" to "Offspring".) Then return to Step 3.

The procedure for moving along the parent 2 direction is the same as that for the parent 1 direction. However, the procedure for the common direction is slightly different from them, due to the different starting points. The starting points from the parents directions are P1 or P2, while the starting point in the common direction is O.

As mentioned earlier, building the common direction depends on whether both parent directions are consistent or not. If they are consistent on ith axis (either both positive or both negative), then move the same direction on the ith axis as the parent directions. Otherwise, stay on that axis without any movement, due to inconsistent directions. There is a special case: one of the moving distances on an axis in the parent directions is zero and the other is nonzero. In this case, we recommend movement in the same direction with the parent direction with nonzero moving distance on the axis.

The procedure for movement along the common direction is as following.

1. Calculate δ_{P1O} and δ_{P2O} as Equations (7) and (8).
2. The next new point is defined as $\mathbf{x}_{New} = [x_{New1}, ..., x_{Newk}]'$ along the path from the common direction. To establish the common direction, three situations on each axis/dimension are possible: (a) the $\delta_{1i} \times \delta_{2i} > 0$ which means that there is a common direction on the ith axis; (b) The $\delta_{1i} \times \delta_{2i} < 0$ which means that there is not a common direction on the ith axis; and (c) the $\delta_{1i} \times \delta_{2i} = 0$ which means that at least one of δ_{1i} and δ_{2i} equals zero.

   2.1. If the situation is (a), then the new point position on the ith axis is given by $x_{Newi} = x_{Oi} + min(|\delta_{1i}|, |\delta_{2i}|, d)$ if both δ_{1i} and δ_{2i} are positive, or $x_{Newi} = x_{Oi} - min(|\delta_{1i}|, |\delta_{2i}|, d)$ if both δ_{1i} and δ_{2i} are negative.

   2.2. If the situation is (b), the new point position on the ith axis is given by $x_{Newi} = x_{Oi}$ (no movement on the ith axis in this situation).

   2.3. If the situation is (c), there are three subcases: (1) $\delta_{1i} = 0$ and $\delta_{2i} \neq 0$; (2) $\delta_{1i} \neq 0$ and $\delta_{2i} = 0$; and (3) $\delta_{1i} = 0$ and $\delta_{2i} = 0$.

      2.3.1. For case (1), if $|\delta_{2i}| \geq d$, then $x_{Newi} = x_{Oi} + d$ (when $\delta_{2i} > 0$) or $x_{Newi} = x_{Oi} - d$ (when $\delta_{2i} < 0$). Otherwise, $x_{Newi} = x_{Oi} + \delta_{2i}$.

      2.3.2. For case (2), similar to case (1), if $|\delta_{1i}| \geq d$, then $x_{Newi} = x_{Oi} \pm d$. Otherwise $x_{Newi} = x_{Oi} + \delta_{1i}$.

      2.3.3. For case (3), $x_{Newi} = x_{Oi}$.

3. Check the range of the new point $\mathbf{x}_{New}$.

4. Evaluate the point $\mathbf{x}_{New}$. If the new point performs worse than the point $\mathbf{x}_O$, then the process for moving along the common direction is stopped. If the new point is better than $\mathbf{x}_O$, then replace the point $\mathbf{x}_{New}$ by the next new point $\mathbf{x}_{New}+\mathbf{\Delta}_{NCO}$, where $\mathbf{\Delta}_{NCO}=\mathbf{x}_{New}-\mathbf{x}_O$. (The "NCO" means "New from Common directions" and "Offspring"). Return to Step 3.

APPENDIX B

Computational Details on A Derivative-based Directional Search by SD

In this appendix, we focus on how to implement SD into the GA process. Suppose that in the i^{th} iteration, the best offspring, which is the best among both the current parent and offspring populations, denoted by $\mathbf{x}_O=[x_{O1},...,x_{Ok}]'$, is found. Then the MGA procedure will implement a direction determined by SD into the GA process.

Based on formula (1), the procedure of building a derivative-based directional search by SD into the GA process between the i^{th} and $(i+1)^{th}$ steps is as follows:

1. The first new point is defined as $\mathbf{x}_1=\mathbf{x}_O-d\nabla f(\mathbf{x}_O)$, where d is the size of a moving distance in each step and $\mathbf{x}_O$ is the best offspring. If $f(\mathbf{x}_1)<f(\mathbf{x}_O)$ in the case of finding a minimum of f, or $f(\mathbf{x}_1)>f(\mathbf{x}_O)$ in the case of finding a maximum, then go to Step 2. Otherwise, the procedure is halted.

2. Compute $\mathbf{x}_{j+1}=\mathbf{x}_j-d\nabla f(\mathbf{x}_j)$, where the iteration index $j=1,2,....$ If $f(\mathbf{x}_{j+1})<f(\mathbf{x}_j)$ for minimization, then repeat Step 2 by letting $j=j+1$. Otherwise, the procedure is halted.

The procedure shows us that it starts at the best offspring, $\mathbf{x}_O$, on the surface of the objective function f and minimizes along the direction of its gradient. This procedure can be improved by using fractional increments (Myers, 1990, page 429) to allow the procedure itself to adjust the moving distance in magnitude to the surface of the objective function. In our study, the strategy on fractional increments implemented into each step of the procedure of building a direction by SD is as follows:

1. Let $l = 1$.
2. Let $\mathbf{x}_{j+1} = \mathbf{x}_j - d\gamma^{l-1}\nabla f(\mathbf{x}_j)$, where γ is a constant value within (0, 1), and $\gamma = 0.5$ in our study.
3. If $f(\mathbf{x}_{j+1}) < f(\mathbf{x}_j)$ for minimization, then the procedure is completed.
4. If $f(\mathbf{x}_{j+1}) > f(\mathbf{x}_j)$, then let $l = l + 1$ and go back to Step 2.
5. If $l > a$ (where a is some constant integer and $a = 5$ in our study), then the procedure is halted.

APPENDIX C

Computational Details on A Derivative-based Directional Search by NR

Implementation by the NR method into a GA process is quite similar to implementation of a search by SD. When the best offspring, which is also the best over the parent population, is found at the i^{th} iteration, implement a directional search by NR into the GA process between the i^{th} and $(i + 1)^{th}$ steps as the following procedure, based on formula (3):

1. The first new point is defined as $\mathbf{x}_1 = \mathbf{x}_O - \mathbf{H}_O^{-1}\nabla f(\mathbf{x}_O)$. If the point $\mathbf{x}_1$ is better than $\mathbf{x}_O$ in terms of f, then go to Step 2. Otherwise, the procedure is halted.
2. Compute $\mathbf{x}_{j+1} = \mathbf{x}_j - \mathbf{H}_j^{-1}\nabla f(\mathbf{x}_j)$, where the iteration index $j = 1, 2, ...$ If $\mathbf{x}_{j+1}$ is better than $\mathbf{x}_j$ in terms of f, then repeat Step 2 by letting $j = j+1$. Otherwise, the procedure is halted.

This procedure can also be improved by using fractional increments as the procedure by SD. The following is the strategy on fractional increments implemented into each step of the procedure of NR.

1. Let $l = 1$.

2. Let $\mathbf{x}_{j+1} = \mathbf{x}_j - \gamma^{l-1}\mathbf{H}_j^{-1}\nabla f(\mathbf{x}_j)$, where γ is a constant value within (0, 1), and $\gamma = 0.5$ in our study.
3. If $\mathbf{x}_{j+1}$ is better than $\mathbf{x}_j$ in terms of f, then the procedure is completed.
4. If $\mathbf{x}_{j+1}$ is worse than $\mathbf{x}_j$ in terms of f, then let $l = l + 1$ and go back to Step 2.
5. If $l > a$ (where a is some constant integer and $a = 5$ in our study), then the procedure is halted.

APPENDIX D

Sphere Model and Schwefel's Function

The sphere model (Schwefel, 1995; Back, 1996; and Haupt and Haupt, 2004) is given by

$$f(\mathbf{x}) = \sum_{i=1}^{k} x_i^2,$$

where the k is the number of dimensions of the function. We chose $k = 2$ in this study and the range is set to $-40 \leq x_i \leq 60$ as in Back (1996). The goal is to find its minimal value and its corresponding location. Obviously, the minimum value is 0 and its location is $(0, 0)$.

A generalized Schwefel's problem 2.26 from Schwefel (1995), is given by

$$\sum_{i=1}^{k} -x_i \sin(\sqrt{|x_i|}), where - 500 \leq x_i \leq 500,$$

where k is the number of dimensions of the function. The minimum of the objective function is given by

$$\min(f(\mathbf{x})) = f(420.9687, ..., 420.9687).$$

The minimum is dependent on k, the number of dimensions. For example, if k =5, then the minimum value is -2094.9144. If k=20, then the minimum value is -8379.6577. Figure 5 shows its 1- and 2-dimensional surfaces.

[Insert Figure 5 about here.]

REFERENCES

Araujo, A. and Assis, F.M. (2000). "An Improved Genetic Algorithm Performance with Benchmark Functions." 292 Electronic Edition.

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

Francisco Ortiz, Jr., Simpson, J.R., Pignatiello, J.J. and Heredia-Langner, A. (2004). "A Genetic Algorithm Approach to Multiple-Response Optimization." *Journal of Quality Technology* 36(4), 432-450.

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

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(1), 28-46.

Hamada M., Martz, H.F., Reese, C.S. and Wilson, A.G. (2001). " Statistical Practice: Finding Near-Optimal Bayesian Experimental Designs via Genetic Algorithms." *The American Statistician* 55(3), 175-181.

Haupt, R.L. and Haupt, S.E. (2004). *Practical Genetic Algorithms* John Wiley and Sons, Inc., New York, NY.

Hinkelmann, K. and Kempthorne, O. (1994). *Design and Analysis of Experiments Volume 1: Introduction to Experimental Design* John Wiley and Sons, Inc., New York, Chichester, Brisbane, Toronto and Singapore.

Holland J.H. (1992). *Adaption in Natural and Artificial Systems: an Introduction Analysis with Applications to Biology, Control, and Artificial Intelligence* A Bradford Book: The MIT Press, Cambridge, Massachusetts, London, England.

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, 199-213.

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

Wolpert, D.H. and Macready, W.G. (1997). “No Free Lunch Theorems for Optimization.” *IEEE Transactions on Evolutionary Computation* 1(1), 76-82.

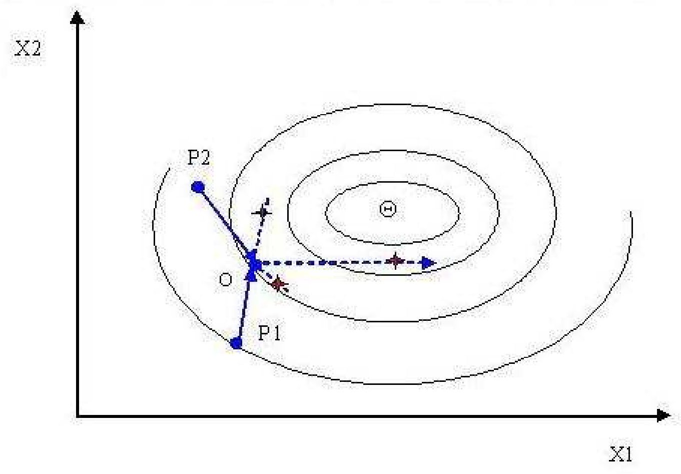

Figure 1: A contour plot of a 2-dimensional problem with the three directions indicated: Parent 1 direction is from P1 to O; Parent 2 direction is from P2 to O; the common direction is a horizontal dotted line, starting at O towards the positive values on the X_1 axis. The three "stars" represent the three points stopped on the three paths with no further improvement.

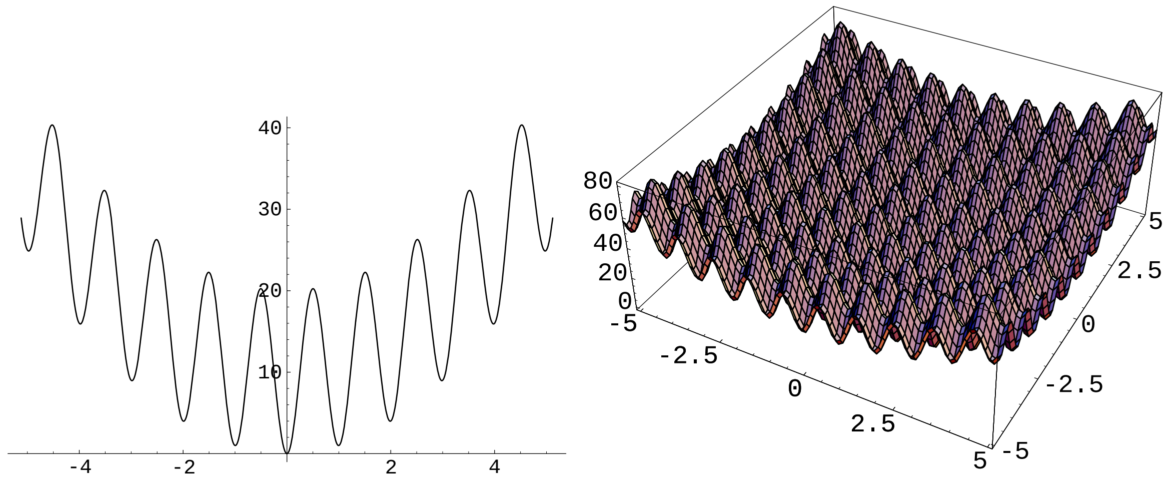

Figure 2: Surface of Rastrigin's function. Left: 1-dimension; right: 2-dimension.

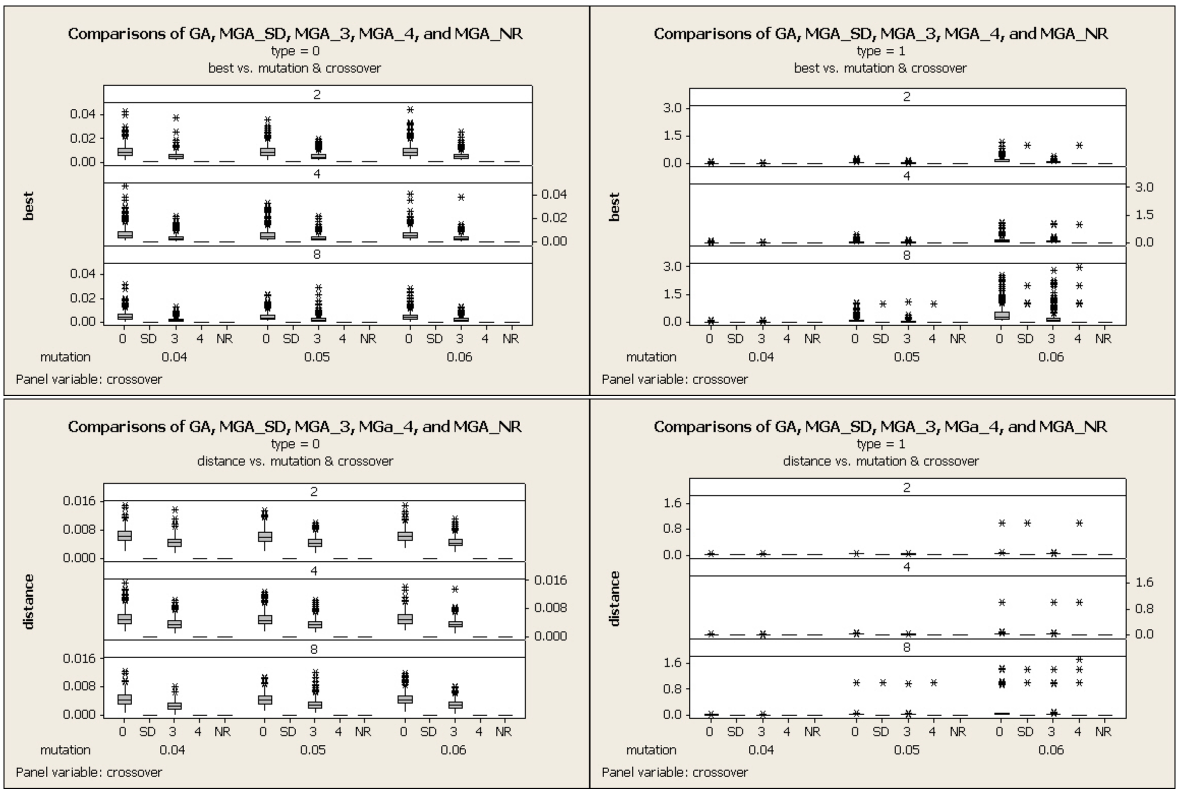

Figure 3: Multiple boxplots for comparisons of GA, MGA$_{SD}$, MGA$_3$, MGA$_4$, and MGA$_{NR}$ (denoted by "0, SD, 3, 4, and NR," respectively) in 18 combinations of the factors type, crossover, and mutation for the Rastrigin's function with 20 dimensions by stopping rule 1: the top left is for the response best when type = 0, the top right is for best when type = 1, the bottom left is for the response distance when type = 0 and the bottom right is for distance when type = 1.

Table 1: Comparisons of GA, MGA$_{SD}$, MGA$_3$, MGA$_4$, and MGA$_{NR}$ (denoted by "0, SD, 3, 4, NR," respectively) in terms of mean of the number of evaluations and the estimated Monte Carlo (MC) error of the mean under the 18 combinations (denoted by "Comb") of the factors type, crossover, and mutation (denoted by "t, c, m," respectively) for the Rastrigin's function in 20-dimensions by stopping rule 2.

Comb			Mean(evaluation)					MC error(mean(evaluation))				
t	c	m	0	SD	3	4	NR	0	SD	3	4	NR
0	2	.04	30706	15792	22266	15925	**115**	331	207	272	208	2
		.05	31310	15859	22585	15996	**114**	339	208	274	204	2
		.06	32366	16415	23274	16600	**113**	327	215	281	218	2
	4	.04	27280	15598	19511	15749	**113**	317	222	270	222	2
		.05	26870	15407	19463	15495	**108**	299	213	254	212	2
		.06	28537	16096	21003	16264	**111**	305	208	262	213	2
	8	.04	25270	15933	17354	16122	**108**	331	227	246	229	2
		.05	25604	16018	17554	16224	**106**	293	218	238	214	1
		.06	26705	16564	18917	16744	**107**	315	227	267	225	2
1	2	.04	51259	31001	36751	30899	**118**	413	371	364	342	2
		.05	77109	45175	53467	45011	**116**	722	548	590	555	2
		.06	118768	69806	81530	70206	**122**	1335	1125	1001	1092	2
	4	.04	45254	31973	33970	31918	**113**	392	338	299	342	2
		.05	70515	49371	53342	49323	**117**	738	627	635	613	2
		.06	113514	78707	87176	78988	**116**	1558	1260	1332	1228	2
	8	.04	46250	37753	37965	38155	**109**	444	429	380	416	2
		.05	89548	74159	71903	73197	**111**	1379	1179	1082	1250	2
		.06	177024	149430	152007	148541	**112**	3482	3237	3327	3356	2

Table 2: Numerical six paired comparisons of GA, MGA$_{SD}$, MGA$_3$, MGA$_4$, and MGA$_{NR}$ (denoted by "0, SD, 3, 4, and NR," respectively) in terms of the number of winners among the 500 replications for each combination with respect to the response evaluation (denoted by "Count(evaluation)") for the Rastrigin's function in 20-dimensions by stopping rule 2. The maximal MC error is 11.

	Count(evaluation)											
	0	SD	0	3	0	4	SD	4	3	4	SD	NR
1	13	**487**	79	**421**	12	**488**	**475**	25	78	**422**	0	**500**
2	8	**492**	66	**434**	11	**489**	**467**	32	72	**428**	0	**500**
3	14	**486**	69	**431**	13	**487**	**477**	23	75	**425**	0	**500**
4	27	**473**	85	**415**	28	**472**	**482**	18	106	**394**	0	**500**
5	25	**475**	80	**420**	25	**475**	**472**	28	107	**393**	0	**500**
6	23	**477**	84	**416**	26	**474**	**480**	20	95	**405**	0	**500**
7	61	**439**	86	**414**	62	**438**	**482**	18	168	**331**	0	**500**
8	49	**451**	65	**435**	53	**447**	**483**	17	187	**313**	0	**500**
9	50	**450**	89	**411**	55	**445**	**483**	17	155	**344**	0	**500**
10	14	**486**	38	**462**	24	**476**	**348**	152	143	**357**	0	**500**
11	25	**475**	53	**447**	25	**475**	**358**	142	149	**351**	0	**500**
12	44	**456**	66	**434**	47	**453**	**367**	133	175	**325**	0	**500**
13	61	**439**	67	**433**	51	**449**	**355**	145	199	**301**	0	**500**
14	65	**435**	93	**407**	68	**432**	**402**	98	204	**296**	0	**500**
15	95	**405**	128	**372**	93	**407**	**409**	91	203	**297**	0	**500**
16	118	**382**	120	**380**	123	**377**	**438**	62	247	**253**	0	**500**
17	158	**342**	150	**350**	149	**351**	**447**	53	243	**257**	0	**500**
18	183	**317**	190	**310**	179	**321**	**435**	65	238	**262**	0	**500**

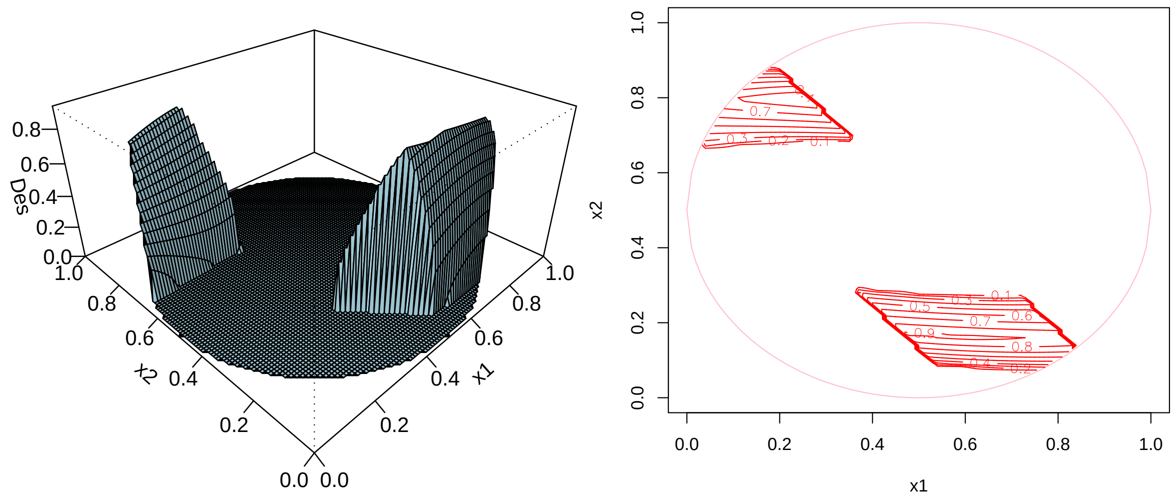

Figure 4: The 3-D surface and the contour of the desirability function (denoted by "Des") within the experimental region R in the case study of a chemical process: left: 3-D surface and right: contour

Table 3: Numerical comparisons of GA, MGA$_{SD}$, MGA$_3$, MGA$_4$, and MGA$_{NR}$ (denoted by "0, SD, 3, 4, NR," respectively) in terms of the MSE of the response best and the estimated MC error of the MSE under the 12 combinations (denoted by "Comb") of the factors type, crossover, and muation (denoted by "t, c, m," respectively) for the case study by stopping rule 1.

Comb			MSE(best) $\times 10^{-3}$					MC error(MSE(best)) $\times 10^{-3}$				
t	c	m	0	SD	3	4	NR	0	SD	3	4	NR
0	0	.4	12.75	11.69	10.02	**8.49**	13.09	2.07	2.07	1.75	1.29	2.14
		.5	7.82	7.21	7.07	**6.64**	8.04	0.86	0.79	0.68	0.62	0.95
		.6	7.14	6.73	6.59	**6.30**	7.22	0.67	0.63	0.54	0.48	0.75
	1	.4	8.99	9.05	8.23	**7.94**	10.42	1.27	1.47	1.39	1.37	1.87
		.5	7.19	7.09	6.58	**6.37**	7.75	0.81	0.85	0.60	0.57	1.10
		.6	6.75	6.50	6.50	**6.34**	6.91	0.63	0.59	0.57	0.56	0.67
1	0	.4	10.18	9.54	8.72	**8.15**	11.37	1.45	1.54	1.34	1.29	1.81
		.5	8.25	7.95	7.22	**6.79**	8.94	0.96	1.26	0.71	0.65	1.38
		.6	7.26	6.62	6.65	**6.31**	7.18	0.71	0.57	0.57	0.48	0.72
	1	.4	8.93	9.56	6.91	**6.76**	10.20	1.43	1.70	0.73	0.70	1.75
		.5	6.53	6.53	6.29	**6.16**	6.77	0.62	0.66	0.51	0.48	0.70
		.6	6.14	6.13	5.95	**5.86**	6.38	0.51	0.52	0.41	0.40	0.57

Table 4: Numerical six paired comparisons of GA, MGA_{SD}, MGA_3, MGA_4, and MGA_{NR} (denoted by "0, SD, 3, 4, and NR," respectively) in terms of the number of winners among the 500 replications for each combination with respect to the response best (denoted by "Count(best)") for the case study by stopping rule 1. The maximal MC error is 11.

	Count(best)											
	0	SD	0	3	0	NR	SD	3	3	4	SD	NR
1	136	**221**	88	**347**	**129**	80	199	**284**	62	**201**	**236**	83
2	145	**247**	134	**307**	**146**	89	229	**253**	65	**215**	**257**	102
3	168	**214**	162	**276**	**157**	84	229	**256**	94	**205**	**227**	119
4	189	**211**	173	**321**	**150**	81	213	**285**	73	**185**	**215**	137
5	177	**224**	199	**293**	**156**	88	242	**255**	74	**176**	**217**	149
6	198	**212**	201	**291**	**160**	88	223	**273**	77	**189**	**246**	139
7	118	**245**	117	**340**	**144**	82	218	**258**	52	**211**	**251**	73
8	127	**264**	145	**317**	**150**	104	244	**247**	70	**228**	**261**	99
9	152	**233**	175	**292**	**136**	94	242	**245**	66	**180**	**247**	109
10	**227**	178	183	**308**	**214**	63	195	**301**	85	**164**	**212**	134
11	**244**	194	235	**261**	**242**	78	217	**280**	100	**169**	**226**	175
12	**263**	181	235	**257**	**223**	80	225	**271**	121	**165**	**223**	186

Table 5: Summary on the GA/MGAs optimal settings (combinations) of the GA operations (type, crossover (denoted by "cross"), and mutation (by "muta")) in all of our examples.

Examples (Functions)	Number factors	Stopping rules	Optimal settings type	cross	muta	Algorithms
Case study	2	1	tour	1	0.6	All
Sphere model "smooth"	2	1	tour	1	0.4	GA
			—	—	—	MGA_{SD}, MGA_4, MGA_{NR}
			tour	1	0.5	MGA_3
		2	tour	1	0.4	GA,MGA_3
			rank	1	0.4	MGA_{SD}, MGA_4, MGA_{NR}
Schwefel's "bumpy"	5	1	tour	3	0.1	All
		2	rank	3	0.2	GA, MGA_3
			rank	2	0.2	MGA_{SD}, MGA_4, MGA_{NR}
	20	1	rank	8	0.05	GA, MGA_3
			tour	8	0.04	MGA_{SD}, MGA_4, MGA_{NR}
		2	rank	8	0.04	GA, MGA_3
			rank	4	0.05	MGA_{SD}, MGA_4, MGA_{NR}
Rastrigin's "very bumpy"	5	1	rank	3	0.2	GA, MGA_{SD}, MGA_3, MGA_4
			—	—	—	MGA_{NR}
		2	rank	3	0.2	GA, MGA_{SD}, MGA_3, MGA_4
			rank	3	0.1	MGA_{NR}
	20	1	rank	8	0.05	GA
			—	—	—	MGA_{SD}, MGA_4, MGA_{NR}
			rank	8	0.04	MGA_3
		2	rank	8	0.04	GA, MGA_3
			rank	4	0.05	MGA_{SD}, MGA_4
			rank	8	0.05	MGA_{NR}

—: Algorithm achieves the optimal solution, zero, in many combinations.
tour: tournament rank: ranking

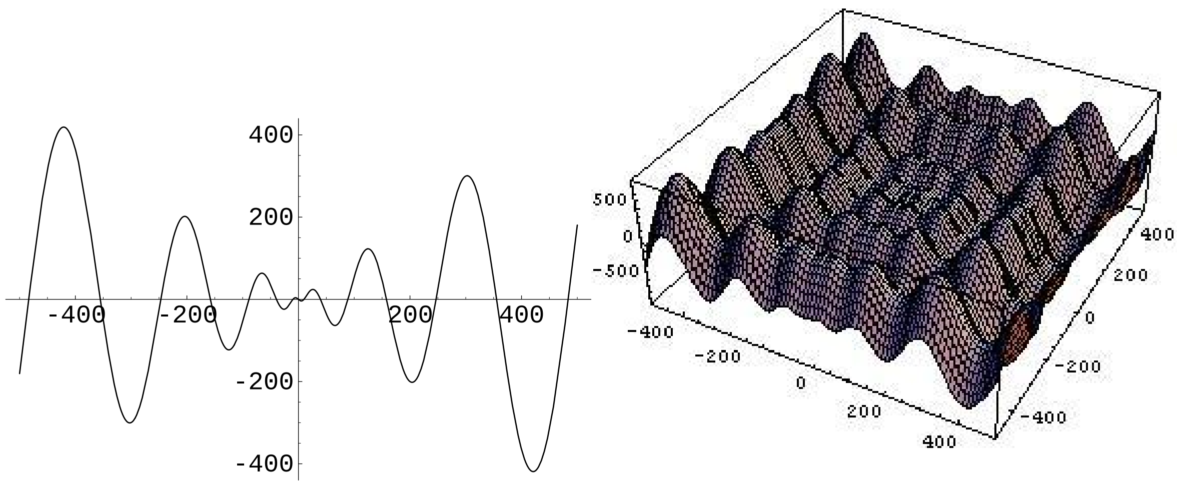

Figure 5: Surface of Schwefel's function. Left: 1-dimension; right: 2-dimension.