*Journal of Global Optimization* **11:** 313–324, 1997.



# A Hybrid Genetic Algorithm for Nonconvex Function Minimization

M. F. HUSSAIN[1] and K. S. AL-SULTAN[2]
[1] *Data Processing Center, King Fahd University of Petroleum and Minerals, Dhahran 31261, Saudi Arabia;* [2]*Department of Systems Engineering, King Fahd University of Petroleum and Minerals, Dhahran 31261, Saudi Arabia* ⋆

(Received 22 June 1995; accepted 27 October 1996)

**Abstract.** In this paper, we consider the problem of minimizing a function in several variables which could be multimodal and may possess discontinuities. A new algorithm for the problem based on the genetic technique is developed. The algorithm is hybrid in nature in the sense that it utilizes the genetic technique to generate search directions, which are used in an optimization scheme and is thus different from any other methods in the literature. The algorithm has been tested on the Rosenbrock valley functions in 2 and 4 dimensions, and multimodal functions in 2 and 4 dimensions, which are of a high degree of difficulty. The results are compared with the Adaptive Random Search, and Simulated Annealing algorithms. The performance of the algorithm is also compared to recent global algorithms in terms of the number of functional evaluations needed to obtain a global minimum and results show that the proposed algorithm is better than these algorithms on a set of standard test problems. It seems that the proposed algorithm is efficient and robust.

**Key words:** Nonconcex function, global optimization, genetic algorithms, search direction, Rosenbrock functions.

## 1. Introduction

Nonlinear programming methods have been used extensively to find the minimum of a given function $f : R^n \rightarrow R$. There are many algorithms in the field of nonlinear programming to find the minimum of $f$, if the function is unimodal. These algorithms are usually classified as either derivative-free or derivative-based methods. The latter methods are more efficient than the former ones but they can only be used if the function is differentiable. These methods, in general, converge to a stationary point (which may not even be a local minimum for functions which are multimodal). For more on these methods see [1–2].

Several techniques have been developed recently for global optimization (Corana et al. [3], and Table III of this paper). Although most of these algorithms perform well in terms of quality of the solution, many of them are very costly in terms of the number of functional evaluations needed to obtain the solution. There are also few algorithms for nonlinear optimization that use genetic techniques [4,

⋆ Address for correspondence; e-mail: asultan@ccse.kfupm.edu.sa

5]. In these algorithms, genetic techniques are used directly on the solution space of the problem and thus are different from our proposed algorithm.

In this paper, a new genetic-based algorithm for optimizing nonlinear multimodal functions in $n$ variables is presented. On the one hand, the algorithm resembles Hooke and Jeeves' algorithm [1] in the sense that it goes through an exploratory search and a pattern search. On the other hand, whereas the Hooke and Jeeves' algorithm uses co-ordinate directions in its exploratory steps, we use genetic techniques to generate random search directions. That is why we classify this algorithm as a hybrid genetic algorithm. The algorithm shares some spirit with the simulated annealing approach [3] in that it overcomes the problem of solution representation, and it is more efficient than existing algorithms because the genetic technique is embedded in an optimization algorithm which uses a line search scheme.

The paper is organized in the following way. In Section 2, we introduce the general methodology of the hybrid genetic algorithm. Section 3 presents the concept of the genetic algorithm and its use in the proposed algorithm. Results and discussion are presented in Section 4.

## 2. Approach

Let $f : R^n \rightarrow R$ be a real valued function. The function $f$ could be multimodal and may be discontinuous. Let $F = \{B^y, 1 \leq y \leq r\}$ be a set of $r$ binary matrices of the form $B^y = (b_{ij}^y), 1 \leq i \leq n; 1 \leq j \leq n_b$, where $n_b$ is the number of bits selected arbitrarily, and $b_{ij}^y = 0$ or 1. An injective mapping $\Gamma$ is used to map the binary matrix $B^y \in F$, to a real vector $d^y \in R^n$. Therefore, it is possible to generate $r$ directions $d^y \in R^n, y = 1, 2, ..., r$ by randomly generating binary matrices $B^y, y = 1, 2, ..., r$ and applying $\Gamma$ on each one of them.

The transformation $\Gamma : F \Rightarrow R^n$ is defined as

$$d^y = \Gamma(B^y) \tag{1}$$

such that

$$d_i^y = (-1)^{b_{i1}^y} \sum_{j=2}^{n_b} 2^{j-2}\, b_{ij}^y, \quad i = 1, 2, \ldots, n. \tag{2}$$

It is clear that the above construction can generate every vector in $R^n$ by appropriate selection of matrices $B$.

In our implementation of the algorithm, we have found that $n_b$ equals 8 works well (i.e. it makes the set of attainable directions in the above lemma dense in $R^n$). Next, we summarize the hybrid genetic approach for function minimization.

The algorithm starts at a random point, say $x_1$ and it goes through several iterations. In any iteration $k$, the algorithm goes through $m$ cycles and one pattern search step. The point $x_k$ of each iteration becomes the starting point $z_1$ for the

$m$ cycles resulting in the points $z_2, z_3, ..., z_{m+1}$ (in each cycle, $r$ directions are generated using the genetic algorithm, and a line search is performed along each direction. The direction that gives the minimum functional value is chosen.) After $m$ cycles, the algorithm takes an optimal step in the direction $z_{m+1} - z_1$ to generate the next point $x_{k+1}$. If $||x_{k+1} - x_k|| \leq \epsilon$, where $\epsilon > 0$, and is sufficiently small, the algorithm stops; otherwise, the algorithm goes through iteration $k + 1$ starting from the point $x_{k+1}$.

Next, we present the general statement of the hybrid genetic algorithm.

## STATEMENT OF THE HYBRID GENETIC ALGORITHM

### *Initialization Step*

Choose a scalar $\epsilon > 0$ to be used for terminating the algorithm. Choose $r$ (the number of random search directions to be used in each cycle), and $m$ (the number of cycles to be performed in each iteration). Choose a starting point $x_1$. Let $z_1 = x_1$. Choose arbitrarily a set of binary matrices $B^i, i = 1, 2, ..., r$. Using the mapping $\Gamma$ in (1), get $d^i, i = 1, 2, ..., r$ corresponding to $B^i, i = 1, 2, ..., r$. For $\ell = 1, 2, ..., r$, let $\lambda^\ell$ be an optimal solution to the problem.

$$\min_{\lambda \in R} f(z_1 + \lambda d^\ell)$$

$$c^\ell = f(z_1 + \lambda^\ell d^\ell).$$

Let $k = j = 1$, let IMAX be the maximum number of iterations, and go to the main step.

### *Main Step*

*1. Perform $m$ cycles.* Let $\lambda_{\text{opt}}$ and $d_{\text{opt}}$ be such that

$$f(z_j + \lambda_{\text{opt}} d_{\text{opt}}) = \min_{1 \leq \ell \leq r} f(z_j + \lambda^\ell d^\ell)$$

(or $d_{\text{opt}}$ is the best direction in this cycle, and $\lambda_{\text{opt}}$ is the optimal step length).

Let

$$z_{j+1} = z_j + \lambda_{\text{opt}}\, d_{\text{opt}}.$$

If $j = m$ go to step 3; otherwise, replace $j$ by $j + 1$ and go to step 2.

*2. Use the genetic routine to generate new search directions*

(a) Use the genetic technique (see Section 3 for details) to generate $r$ new binary matrices $B^i, i = r+1, r+2, ..., 2r$ using the current matrices $B^i, i = 1, 2, ..., r$.

(b) Using the mapping $\Gamma$ in equations (1) and (2), get directions $d^i, i = r+1, r+2, ..., 2r$. Let $\ell = r+1$.
(c) Let $\lambda^\ell$ be an optimal solution to the problem.
$$\min_{\lambda \in R} f(z_j + \lambda d^\ell).$$
Let $c^\ell = f(z_j + \lambda^\ell d^\ell)$. If $\ell < 2r$, replace $\ell$ by $\ell + 1$, and repeat this step.
(d) Pick the minimum $r$ of $c^\ell, \ell = 1, 2, ..., 2r$; rename them as $c^1, c^2, ..., c^r$, and their corresponding binary matrices $B^1, B^2, ..., B^r$, and return to step 1.

*3. Perform pattern search.* Let $d = z_{m+1} - z_1$. Let $\hat{\lambda}$ be an optimal solution to the problem

$$\min_{\lambda \in R} f(z_{m+1} + \lambda d).$$

Let $x_{k+1} = z_{m+1} + \hat{\lambda} d$, and go to step 4.

*4. Check termination criteria.* If $||x_{k+1} - x_k|| < \epsilon$, or $k = IMAX$, stop; otherwise, let $j = 1$, $z_1 = x_{k+1}$, replace $k$ by $k + 1$, and go to step 2.

REMARK. The proposed algorithm uses two parameters, namely $r$ (the number of random searches per cycle), and $m$ (the number of cycles per iteration.) The value of $r$ affects the intensity of the search, i.e., the larger the value of $r$ the more the neighborhood of the current point is searched (i.e. the search is intensified around this point.) The value of $m$ affects the diversification of the search. A large value of $m$ means that the algorithm explores different regions more. Hence the ratio of $m$ to $r$ represents the ratio between diversification and intensification of the search. $m$ should be greater than or equal to $n$, and $r$ should be greater than or equal to 2.

## CONVERGENCE OF THE PROPOSED ALGORITHM

Suppose that $f$ is differentiable, and let the solution set $\Gamma = \{\bar{x} : \nabla f(\bar{x}) = 0\}$. Assume also that $m = n$, or that the number of cycles is equal to the dimension of the space. Assume also that the optimum directions $d_{\text{opt}}$ found in Step 1 are linearly independent. Now, each iteration of the algorithm consists of $n$ searches along linearly independent directions (where, we define the direction here to be $d_{\text{opt}}$ found in each cycle), and a pattern search. Denote the first $n$ searches by the map $B$, and the pattern search by the map $C$. Using an argument similar to standard theorems that establish convergence of algorithms using linearly independent and orthogonal search directions to stationary points (e.g. Theorem 7.3.5 of Bazaraa et al. [1, pp. 254–255]), it follows that $B$ is closed. If the minimum of $f$ along any line is unique and letting $\alpha = f$, then $\alpha(x_{k+1}) < \alpha(x_k)$ for $x_k \notin \Gamma$. By definition of C, $\alpha(z) \leq \alpha(x_{k+1})$, for $z \in C(x_{k+1})$. Assuming $\wedge = \{x : f(x) \leq f(x_1)\}$, where $x_1$ is the starting point, is compact, convergence of the proposed algorithm can

be established using standard theorems that establish convergence of algorithms with composite maps to stationary points (e.g. theorem 7.3.4 of Bazaraa et al. [1, p. 253]). ■

One can notice that the proof of the procedure stated above requires a unique minimizer over all lines in $R^n$, making the functions strictly quasiconvex and hence the local solutions are also global solutions. Therefore, the above proof does not stand for general nonlinear functions. However, one can appeal to a spacer step to guarantee theoretical convergence. This can be achieved by inserting such a spacer step involving a periodic minimization along the negative gradient direction (which always exists due to the differentability assumption on the function). Clearly, these spacer steps are steps of the steepest descent algorithm which is known to converge, then the only requirement for the whole procedure to converge is that the other steps of the procedure do not increase the value of the function which is fulfilled by our procedure (see spacer step theorem in Luenberger [6, p. 231] as well as discussion on spacer steps in Bazaraa et al. [1, p. 253, and p. 237]).

One should note that it might be expensive to check linear independence for directions especially if the dimension of the space is large. One might either ignore this hoping that it will be rare that randomly generated directions will be linearly dependent, or one can generate randomly directions of the form $d(\delta_i)$, where $\delta_i = 1, 0$ or $-1$ and restrict the selection of directions in each cycle to guarantee linear independence (this latter idea makes the genetic algorithm operate directly on the directions rather than on the binary matrices. One clear disadvantage of this is that the directions generated using this scheme are rather limited compared to the original scheme.) At this point, the following comments are in order:

- As it is clear from the statement of the algorithm, $d_{\text{opt}}$ is the best direction among the $r$ directions used in each cycle, and it is the only direction used for updating the point $z_j$ at the end of the cycle (the remaining $r - 1$ directions are only used for exploratory purposes, and therefore, they are not used to update the point).
- As stated earlier the proposed algorithm can be established to converge to a stationary point provided that some conditions are satisfied, which is the case for most of the classical nonlinear programming algorithms. However, it is conjectured that the proposed algorithm will not get stuck in such points, and it will eventually converge to a global minimum. This conjecture is supported by the good though limited empirical results presented in Section 4 of this paper. However, it remains to be proven.

## 3. The General Genetic Technique

Genetic algorithms are basically search techniques. They emulate the natural process of evolution while progressing towards the optimum. These algorithms have been used extensively in solving various optimization problems [7–8].

In a genetic approach, at any given instant of time, a population of possible solutions is generated. Their number is arbitrarily chosen, but somehow depends on the nature and size of the problem. It may also depend on the memory size of the computer on which it is being implemented. Each element of the population is called a chromosome, which is a combination of symbols, known as genes. Chromosomes are possible solutions to the problem. In each generation, the best chromosomes are selected using the Roulette principle, to act as new parents (the Roulette principle is explained next). Three genetic operators known as reproduction, crossover, and mutation are applied to these parents to generate new offspring. These new offspring inherit good qualities from the parents. In the usual genetic algorithm, the process of generating new offspring is continued until there is no further improvement in the offspring [6]. However, in our case, we use the generated offspring in an optimization algorithm. Next, we discuss a scheme for performing the Roulette principle.

### THE ROULETTE PRINCIPLE

Assume that the current point is $x_1 \in R^n$, and that one has $r$ binary matrices at hand. The transformation $\Gamma$ is applied on all matrices $B^i$, $i = 1, 2, ..., r$, which constructs $r$ directions $d^i \in R^n$, $i = 1, 2, ..., r$, as defined in eq. (1) and eq. (2). Then let

$$c^i_{\min} = \min_{\lambda \in R} f(x_1 + \lambda d^i), \quad \forall d^i, \quad i = 1, \dots r \tag{3}$$

be the cost associated with each direction $d^i$. Assume, without loss of generality, that the costs in (3) are ordered in an ascending order or $c^1_{\min} \leq c^2_{\min} \dots \leq c^r_{\min}$, and let $B^1, B^2, ..., B^r$ and $d^1, d^2, ..., d^r$ be the corresponding binary matrices and directions respectively. Compute $\Delta^i = \left[c^r_{\min} + 1 + \frac{1}{r}\right] - c^i_{\min}, i = 1, 2, ..., r$, and generate $p = (p_1, \dots, p_i, \dots, p_r)$ using the following formula

$$p_i = \frac{\sum_{k=1}^{i} \Delta^k}{\sum_{k=1}^{r} \Delta^k} \quad , \quad i = 1, 2, ..., r. \tag{4}$$

Now, to pick a parent out of the current population, a scheme is needed which assigns higher probabilities of selection to parents with smaller objective functions values than to those parents with worse objective function values. We propose the following scheme which achieves the above requirement. Calculate $p_i, i = 1, 2, ..., r$, using equation (4) above. Generate a random number $v \sim u(0, 1)$, where $u(0, 1)$ is the uniform distribution between 0 and 1. A cut $s$ is selected if $p_{s-1} \leq v \leq p_s$, where $p_0 = 0$. This makes $d^s$ the favorite direction of search, which, in turn makes $B^s$ the favorite parent (i.e. the parent to be picked).

The above suggested scheme for performing the Roulette principle can be used to select parents that are necessary for performing the reproduction, crossover, and

mutation operators. For more detail on these operators see [7–8]. One can select each one of the above operators to be executed with certain predefined probabilities.

## 4. Results and Discussion

The algorithm developed in this paper has been tested on various functions, with different orders of difficulty. Two sets of test functions are used which are the same as those used by Corana et al. [3] to facilitate comparison.

The first set consists of the Rosenbrock valley functions in 2 and 4 dimensions. These are classical examples of ill-conditioned, unimodal functions. The Rosenbrock valley functions can be found in [3] and [9]. The second set consists of multimodal functions in 2 and 4 dimensions as constructed by Corana et al. [3]. These functions are basically paraboloid with axes parallel to co-ordinate directions. They contain a very high number of minima, and possess strong discontinuities. Such functions quickly trap any unimodal optimization algorithm at a local minimum. This property makes them a very tough test for any optimization algorithm. By testing our algorithm on the same functions used by Corana et al. [3], we can compare our algorithm with the Simulated Annealing method [3], and the Adaptive Random Search method [10]. As stated earlier, the functions used for testing are of a high degree of difficulty. Thus, there is a high probability of the algorithms being trapped at local minima.

The tests were performed starting from different initial points, with the initial directions generated randomly. Line searches were performed using the Golden Section technique with tolerance $10^{-6}$. $m = 3n + 1$ was used. $r = 7$ for $n = 2$ and $r = 12$ for $n = 4$ were selected after some preliminary testing. These tests were carried out on a PC–386 with a co-processor. The time taken to perform these tests was much less than that reported by Corana et al. [3]. However, we rely on the number of function evaluations as our measure of efficiency, because it is machine-independent.

We have presented the results obtained for the Rosenbrock functions in 2 and 4 dimensions in Table I. For comparison purposes, the results for the Adaptive Random Search, and Simulated Annealing algorithms are taken from [3]. The results for the parabolic multiminima functions in 2 and 4 dimensions are presented in Table II. The results for the Simulated Annealing algorithm are also presented for comparison. It can be seen from the tables that the hybrid genetic algorithm always converges to the global optimal value irrespective of the initial point. In comparison, the Simulated Annealing algorithm ends up with local minima in a few cases [3], though they are very close to the optimal solution. The Adaptive Random Search algorithm stops at saddle points and fails to converge to the optimal values in some instances [3].

The efficiency of the algorithms is measured by the number of function evaluations. It can be observed that the hybrid genetic algorithm is much more efficient than the Simulated Annealing algorithm. The function evaluations needed are

Table I. Comparison of the genetic algorithm with other methods for Rosenbrock valley functions in 2 and 4 dimensions

| Starting Point | Method | | | | | |
|---|---|---|---|---|---|---|
| | Function Evaluations | | | Final Function Value | | |
| | *ARS | *SA | GA | *ARS | *SA | GA |
| 2 Dimensions: | | | | | | |
| 1001,1001 | 3411 | 500001 | 2389 | 1586.4 | 1.8E-10 | 1.2E-12 |
| 1001,−999 | 131841 | 508001 | 2214 | 8.6E-9 | 2.6E-9 | 2.3E-10 |
| −999,−999 | 15141 | 524001 | 3254 | 1.2E-8 | 1.2E-9 | 4.4E-11 |
| −999,1001 | 3802 | 484001 | 3412 | 583.2 | 4.2E-8 | 3.4E-10 |
| 1443,1 | 181280 | 492001 | 2115 | 4.7E-10 | 1.5E-8 | 3.3E-10 |
| 1,1443 | 2629 | 512001 | 5781 | 1468.9 | 1.6E-9 | 1.2E-10 |
| 1.2,1 | 6630 | 488001 | 1548 | 5.5E-7 | 2.0E-8 | 2.2E-18 |
| 4 Dimensions | | | | | | |
| 101,101,101,101 | 519632 | 1288001 | 228534 | 1.9E-6 | 5.0E-7 | 4.8E-9 |
| 101,101,101,−99 | 194720 | 1328001 | 213422 | 1.7E-6 | 1.8E-7 | 2.1E-9 |
| 101,101,−99,−99 | 183608 | 1264001 | 264521 | 3.8E-6 | 5.9E-7 | 2.2E-8 |
| 101,−99,−99,−99 | 195902 | 1296001 | 299321 | 2.3E-6 | 7.4E-8 | 3.0E-9 |
| −99,−99,−99,−99 | 190737 | 1304001 | 44567 | 2.7E-6 | 3.3E-7 | 5.7E-9 |
| −99,101,−99,101 | 4172290 | 1280001 | 234512 | 2.6E-6 | 2.8E-7 | 3.9E-8 |
| 101,−99,101,−99 | 53878 | 1272001 | 193134 | 3.7 | 2.3E-7 | 4.1E-8 |
| 201,0,0,0 | 209415 | 1288001 | 182131 | 1.1E-6 | 7.5E-7 | 3.0E-8 |
| 1,201,1,1 | 215116 | 1304001 | 283946 | 1.2E-6 | 4.6E-7 | 5.8E-8 |
| 1,1,1,201 | 29069006 | 1272001 | 214312 | 2.2E-6 | 5.2E-7 | 4.3E-8 |

** Results taken from Corana et al. [3].

approximately 20% of those required by the Simulated Annealing algorithm with equal or better accuracy. It is also important to note that apart from a few exceptions, the number of function evaluations needed to find an optimum does not vary much for different initial points.

We have also compared the performance of the proposed algorithm with some of the recent algorithms in the literature which are shown in Table III. Table IV gives the number of function evaluations taken by methods A-H for the set of test functions proposed by Dixon and Szegö [18] (see Table VI). In Table V, the running times in units of standard time for these methods are given, where one unit of standard time is defined to be the running time for 1000 evaluations of the Shekel 5 function at the point $(4, 4, 4, 4)$ (see [18]). There are no times for methods G, since the reported times for G are in absolute computer time and not available in standard time units as for other methods. Results for methods A-G are taken from Dekkers and Arts [16].

Clearly, our proposed algorithm requires less number of function evaluations, and less execution time than the other methods in most of the cases. One should

Table II. Comparison of the genetic algorithm with simulated annealing for parabolic, multiminima functions in 2 and 4 dimensions

| Starting Point | Method | | | |
|---|---|---|---|---|
| | Function Evaluations | | Final Function Value | |
| | **SA | GA | **SA | GA |
| 2 Dimensions: | | | | |
| 100,888 | 684000 | 1814 | 2.5E-8 | 3.4E-9 |
| −999,1001 | 680000 | 2859 | 3.2E-9 | 2.6E−10 |
| −999,−889 | 708000 | 2438 | 4.0E-9 | 8.7E-11 |
| 1001,−998 | 696000 | 2674 | 1.2E-9 | 2.9E-10 |
| 1441,3 | 708000 | 2934 | 3.2E-9 | 5.4E-11 |
| −10,−1410 | 680000 | 2412 | 4.0E-8 | 4.6E-9 |
| −1100,850 | 696000 | 2661 | 1.2E-8 | 3.2E-9 |
| 850,−1100 | 656000 | 2345 | 4.2E-10 | 6.0E-11 |
| 4 dimensions: | | | | |
| −999,−999,−999,−1000 | 1440000 | 230002 | 2.6E-7 | 2.9E-8 |
| 999,1000,1001,−998 | 1160000 | 210331 | 3.4E-3 | 3.3E-6 |
| 1000,−1000,10000,−1000 | 1464000 | 143135 | 8.7E-8 | 7.4E-10 |
| −999,−999,−998,−1000 | 1440000 | 185439 | 2.0E-7 | 1.8E-9 |
| 1000,999,999,998 | 1424000 | 236231 | 3.4E-7 | 8.4E-10 |
| 1000,−1000,−9999,9999 | 1416000 | 164548 | 2.5E-7 | 1.1E-9 |
| 1000,−1000, 998,1000 | 1176000 | 214583 | 3.4E-3 | 3.1E-6 |
| 0,0,1,2001 | 1408000 | 223121 | 4.0E-7 | 3.8E-8 |
| 1998,3,10,−13 | 1408000 | 173154 | 4.6E-7 | 1.5E-8 |
| 1234,−1234,560,−334 | 1432000 | 204562 | 3.4E-7 | 5.3E-9 |

** Results taken from Corana et al. [3].

Table III. Methods used in our second comparison

| Method | Name | Reference |
|---|---|---|
| A | Multistart | Rinnooy Kan and Timmer [11] |
| B | Controlled Random Search | Price [12] |
| C | Density Clustering | Törn [13] |
| D | Clustering with distribution function | De Biase and Frontini [14] |
| E | Multi Level Single Linkage | Rinnooy Kan and Timmer [15] |
| F | Simulated Annealing | Dekker and Aarts [16] |
| G | Simulated Annealing based on stochastic differential equations | Aluffi-Pentini et al. [17] |
| H | Hybrid Genetic Algorithm | This paper |

also notice that the multilevel single linkage requires 1000 function evaluations for the random sample, and the simulated annealing of Dekkers and Arts requires 10 $n$ function evaluations for initialization, where $n$ is the dimension and these are not

Table IV. Number of function evaluations needed for methods in Table III for the functions shown in Table VI.

| Function | GP | BR | H3 | H6 | S5 | S7 | S10 |
|---|---|---|---|---|---|---|---|
| | 2 | 2 | 3 | 6 | 4 | 4 | 4 |
| A | 4400 | 1600 | 2500 | 6000 | 6500 | 9300 | 11000 |
| B | 2500 | 1800 | 2400 | 7600 | 3800 | 4900 | 4400 |
| C | 2499 | 1558 | 2584 | 3447 | 3649 | 3606 | 3874 |
| D | 378 | 597 | 732 | 807 | 620 | 788 | 1160 |
| E | 148 | 206 | 197 | 487 | 404 | 432[a] | 564 |
| F | 563 | 505 | 1459 | 4648 | 365[a] | 558 | 797 |
| G | 5439 | 2700 | 3416 | 3975 | 2446 | 4759 | 4741 |
| H[b] | 146 | 199 | 191 | 482 | 403 | 521 | 559 |

[a] The global minimum was not found in one of the four runs.
[b] Average of four runs.

Table V. Running times in units of standard time for methods shown in Table III for functions given in Table VI

| Function | GP | BR | H3 | H6 | S5 | S7 | S10 |
|---|---|---|---|---|---|---|---|
| | 2 | 2 | 3 | 6 | 4 | 4 | 4 |
| A | 4.5 | 2 | 7 | 22 | 13 | 21 | 32 |
| B | 3 | 4 | 8 | 46 | 14 | 20 | 20 |
| C | 4 | 4 | 8 | 16 | 10 | 13 | 15 |
| D | 15 | 14 | 16 | 21 | 23 | 20 | 30 |
| E | 0.15 | 0.25 | 0.5 | 2 | 1 | 1[a] | 2 |
| F | 0.9 | 0.9 | 5 | 20 | 0.8[a] | 1.5 | 2.7 |
| H[b] | 0.15 | 0.24 | 0.49 | 1.99 | 1.00 | 1.25 | 1.99 |

[a] The global minimum was not found in one of the four runs.
[b] Average of four runs.

Table VI. Test functions

| | | |
|---|---|---|
| GP | Goldstein and Price | Dixon and Szegö [18] |
| BR | Branin | Dixon and Szegö [18] |
| H3,H6 | Hartmann's family | Dixon and Szegö [18] |
| S5,S7,S10 | Shekel's family | Dixon and Szegö [18] |

shown in Tables IV and V. This means that our algorithm is actually much more efficient than these algorithm as all function evaluations needed for our algorithm are reported in Table IV. Moreover, the proposed algorithm is more reliable than some of these algorithms as it never failed to get the global solution in any of the runs, while some of these methods did fail in few of the runs as shown in Table IV.

## 5. Conclusions

In this paper, a new approach for the optimization of multimodal functions has been presented. The method is based on constructing search directions using genetic techniques. This overcomes the problem associated with representation of real solutions. The algorithm has been tested on several standard test problems. It has proven to be very robust and requires fewer function evaluations than recent global algorithms in the literature.

## Acknowledgement

The authors are grateful to anonymous referees for many comments that have improved the paper. The authors would also like to acknowledge the support provided by King Fahd University of Petroleum and Minerals for conducting this research.

## References

1. Bazaraa, M.S., H. Sherali, and C.M. Shetty, *Nonlinear Programming: Theory and Algorithms*, second edition, John Wiley and Sons, New York, 1993.
2. Reklaitis, G.V., A. Ravindran, and K.M. Ragsdell, *Engineering Optimization: Methods and Applications.* John Wiley, New York, 1983.
3. Corana, A., M. Marchesi, C. Martini, and S. Ridella, "Minimizing multimodal functions of continuous variables with the Simulated Annealing algorithm", *ACM Trans. Math. Softw.* 13(3), 1987, 262–280.
4. Hajela, P., "Genetic search – An approach to the nonconvex optimization problem", *AIAA J.* 28(7), 1990, 1205–1210.
5. Pham, D.T., and Y. Yang, "Optimization of multi-modal discrete functions using genetic algorithms", *Proc. Instn. Mech. Engrs.* 207, 1993, 53–59.
6. Luenberger, D.G., *Linear and Nonlinear Programming,* Addison Wesley, Second Edition, USA, 1984.
7. Goldberg, D.E., *Genetic Algorithms in Search, Optimization, and Machine Learning.* Reading Mass, Addison-Wesley, 1989.
8. Lawrence, D. (Ed.), *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, New York, 1991.
9. Rosenbrock, H., "An automatic method for finding the greatest or least value of a function", *Comput. J.* 3, 1960, 175–184.
10. Masri, S.F., G.A. Bekey, and F.B. Safford, "A global optimization algorithm using adaptive random search", *Appl. Math. Comput.* 7, 1980, 353–375.
11. Rinnooy Kan, A.H.G., and G.T. Timmer, "Stochastic methods for global optimization", *American Journal of Mathematics and Management Sciences* 4, 1984, 7–40.
12. Price, W.L., "A controlled random search procedure for global optimization", in : L.C.W. Dixon and G.P. Szegö (Eds.), *Towards Global Optimization 2,* North Holland, Amsterdam, 1978, pp. 71–84.

13. Törn, A.A., "A search-clustering approach to global optimization", in L.C.W. Dixon and G.P. Szegö (Eds.), *Towards Global Optimization 2*, North Holland, Amsterdam, 1978, pp. 49–62.
14. De Biase, L., and F. Frontini, "A stochastic method for global optimization: its structure and numerical performance", in L.C.W. Dixon and G.P. Szegö (Eds.), *Towards Global Optimization 2*, North-Holland, Amsterdam, 1978, pp. 85–102.
15. Rinnooy Kan, A.H.G., and G.T. Timmer, "Stochastic global optimization methods. Part II: multi level methods", *Mathematical Programming* 39, 1987, 57–78.
16. Dekkers, A., and E. Aarts, "Global optimization and simulated annealing", *Mathematical Programming* 50, 1991, 367–393.
17. Aluffi-Pentini, F., V. Parisi, and F. Zirilli, "Global optimization and stochastic differential equations", *Journal of Optimization Theory and Applications* 47, 1985, 1–16.
18. Dixon, L.C.W., and G.P. Szegö, "The global optimization problem: an introduction", in : L.C.W. Dixon and G.P. Szegö (Eds.), *Towards Global Optimization 2*, North Holland, Amsterdam, 1978, pp. 1–15.