Real-coded genetic algorithm with variable rates of cross-over and mutation: a basis of global optimization for multi-modal functions via interval technique

S. D. JABEEN* and A. K. BHUNIA*

Department of Mathematics, University of Burdwan, Burdwan-713104, India

(Received April 2005; revised version received 2 June 2006; second revision received 21 September 2006; third revision received 29 October 2006; accepted 8 November 2006)

A hybrid method consisting of a real-coded genetic algorithm (RCGA) and an interval technique is proposed for optimizing bound constrained non-linear multi-modal functions. This method has two different phases. In phase I, the search space is divided into several subregions and the simple genetic algorithm (SGA) is applied to each subregion to find the one(s) containing the best value of the objective function. In phase II, the selected subregion is divided into two equal halves and the advanced GA, i.e. the RCGA, is applied in each half to reject the subregion where the global solution does not exist. This process is repeated until the interval width of each variable is less than a pre-assigned very small positive number. In the RCGA, we consider rank-based selection, multi-parent whole arithmetical cross-over, and non-uniform mutation depending on the age of the population. However, the cross-over and mutation rates are assumed as variables. Initially, these rates are high and then decrease from generation to generation. Finally, the proposed hybrid method is applied to several standard test functions used in the literature; the results obtained are encouraging. Sensitivity analyses are shown graphically with respect to different parameters on the lower bound of the interval valued objective function of two different problems.

Keywords: Global optimization; Genetic algorithm; Interval technique; Multi-modal continuous function

1. Introduction

In many practical situations, algorithms for solving optimization problems are of increasing importance with respect to global perspective. Generally, these problems are non-convex (or non-concave), multi-modal, and high dimensional with numerous local optima. In solving such types of problems, finding a global solution is a formidable task. As a result, several stochastic search optimization methods have been suggested to search for a global optima. They are usually heuristic in nature and very expensive to apply with respect to the CPU time required. Therefore, different types of hybrid algorithms are becoming popular. Generally, these hybrid methods are developed using a combination of evolutionary algorithms and...
some other methods. Evolutionary algorithms are based on the principles of evolution and heredity. Among these, genetic algorithms (GAs), which are search and optimization algorithms based on the mechanics of natural selection and natural genetics [1], have become very popular. These algorithms start from a randomly generated population of individuals, each of which is generally represented as a one-dimensional string of values, called genes, in binary/octal/hexadecimal coding. However, in real coding, each individual is encoded as a vector of floating-point numbers with the same components as the vector of decision variables of the problems. These individuals represent the solution of the problem. The higher the fitness of the individual, the larger (smaller) is the value of the objective function being maximized (minimized). The fitness is a value given by the function to be optimized, which measures the quality of solution represented by the individual. GAs seek to find good solutions using analogues of natural selection together with mutation and sexual reproduction. The most frequently used such operators are reproduction/selection, cross-over, and mutation. The reproduction/selection operator selects fitter individuals from the population to generate the offspring. The cross-over operator combines the features of two or more randomly selected parent chromosomes in order to generate new individuals. Finally, the mutation operator generates offspring by randomly changing one or several genes in an individual/chromosome. Thus offspring may possess different characteristics from their parents. This operator inhibits concentration exclusively on local search of the search space and increases the chance of finding global optima. For this reason, this algorithm has received considerable attention regarding its potential as a novel optimization technique for complex problems and has been successfully used in a wide variety of problems in business, engineering, and the sciences.

Recently, Chelouah and Siarry [2] developed a continuous GA for global optimization of multi-modal functions, and Bessaou and Siarry [3] proposed a real-coded GA to optimize the multi-modal continuous function using entropy theory. Hussain and Al-Sultan [4] proposed a hybrid GA for minimizing a non-convex multi-modal function and compared their results with the results obtained using adaptive random search and simulated annealing algorithms. Using multi-mutation operators, Hong et al. [5] developed a dynamic mutation GA which shows better performance than other GAs with single mutation operator. Yiu et al. [6] proposed a hybrid descent method consisting of a simulated annealing algorithm and a gradient-based method. Storn and Price [7] presented an efficient heuristic approach for minimizing non-linear and non-differential continuous functions using differential evolution. Barrios et al. [8] proposed another approach to optimize a function by using a real-coded GA based on mathematical morphology theory. Very recently, Salhi and Queen [9] developed a hybrid approach that combines simulated annealing, tabu search, and a descent method. Using this method, they found all the global minima and some local minima in all the cases they examined. Ali and Törn [10] studied the efficiency and robustness of some well known population-set-based direct search global optimization methods such as controlled random search, differential evolution and the GA. Considering permutation property Wang and Wu [11] developed a hybrid GA by integrating both the evolutional and neighbourhood search. However, for a large search space and multi-modal non-linear multi-dimensional functions (obviously non-convex) there is no guarantee of 100% success rates with respect to the global perspective using of the methodologies listed above.

Therefore we propose a hybrid method of optimizing a bound constrained non-linear multi-modal continuous function with several variables under the global perspective. This method has two different phases. In phase I, single (or multiple) good solutions are found using a simple GA (SGA) by breaking down the search space into smaller subregions. The subregions exhibiting better solutions are extracted and passed to phase II. The idea of recursive subdivision is to concentrate the computational effort into smaller subregions (intervals) as the precision of the solution is progressively increased. This helps to prevent failure of the algorithm due
Global optimization for multi-modal functions

855

to premature loss of diversity. In phase II, the selected subregion is divided into two equal halves and the advanced GA, i.e., the real-coded GA (RCGA), is used to reject the subregion where the global optimum seems less likely. The process is repeated with respect to each variable, reducing the search space consecutively until the interval width of each variable is less than a pre-assigned very small positive number. Finally, this method is tested on several standard test functions available in the literature; the results obtained are encouraging. The sensitivity analyses are shown graphically and statistically with respect to different parameters on the lower bounds of the interval-valued objective function of two different minimization problems.

2. Interval technique in optimization

We consider a bound constrained optimization (minimization or maximization) problem

\[ f = f(x_1, x_2, \ldots, x_n), \quad l_j \leq x_j \leq \bar{l}_j, \quad j = 1, 2, \ldots, n \] (1)

where \( n \) represents the number of variables and \( x_j \) is the \( j \)th variable whose lower and upper bounds are \( l_j \) and \( \bar{l}_j \), respectively. The search space of (1) is defined as

\[ D = \{ x \in \mathbb{R}^n : l_j \leq x_j \leq \bar{l}_j, \quad j = 1, 2, \ldots, n \}. \]

In phase I, the prescribed search space \( D \) has been broken into smaller subspaces \( D_i (i = 1, 2, \ldots, m) \) such that

\[ D = \bigcup_{i=1}^{m} D_i, \quad m \in I^+. \]

In each subregion \( D_i \), the simple GA is applied to investigate those subregions most likely to contain an optimal, or near-optimal, value of \( f \) (the function to be optimized). Division of the search space is basically done to maintain the genetic diversity, as loss in diversity leads to premature convergence and failure to achieve global optima. The procedure for the SGA is given below.

3. The simple genetic algorithm

Step 1: Construct an initial population of a finite number \( N \) of individuals for evolution.
Step 2: Compute the fitness function for the individuals.
Step 3: Upgrade the population with the genetic operators selection, cross-over, and mutation to generate offspring.
Step 4: Compute the fitness of each individual.
Step 5: Select a certain number of best individuals according to their fitness.
Step 6: If the termination condition is satisfied, stop the algorithm; otherwise go to step 4.

In phase II, we extract from phase I those subspaces which have provided the best value of the objective function, as well as the subspaces containing the objective function values.
closest to the best value. Each of these subspaces is defined as follows:

\[ R = \{ x \in \mathbb{R}^n : l_j \leq x_j \leq \bar{l}_j, j = 1, 2, \ldots, n \}. \]

\( R \) can then be divided into two subregions \( R_1 \) and \( R_2 \) with respect to the variable \( x_j (j = 1, 2, \ldots, n) \) as defined below:

\[ R_1 = \{ x \in \mathbb{R}^n : l_i \leq x_i \leq (l_i + \bar{l}_i)/2, l_j \leq x_j \leq \bar{l}_j, j = 1, 2, \ldots, i-1, i+1, \ldots, n \} \]

\[ R_2 = \{ x \in \mathbb{R}^n : (l_i + \bar{l}_i)/2 \leq x_i \leq \bar{l}_i, l_j \leq x_j \leq \bar{l}_j, j = 1, 2, \ldots, i-1, i+1, \ldots, n \} \]

Let \( f_1 \) and \( f_2 \) be the best found values of \( f(x) \) obtained from RCGA in phase II on the subregions \( R_1 \) and \( R_2 \), respectively. The selection of a subregion which is likely to contain a good solution and the rejection of a subregion containing a poor solution can be done by comparing \( f_1 \) and \( f_2 \).

The entire process is repeated until the domain reduces to an interval of negligible width. In fact, we finally obtain the values of different variables in the form of intervals with negligible width. These interval values of different variables will be the best found solution of (1).

4. The advanced algorithm

**Algorithm 1**

Step 1. Initialize the number \( n \) of variables and their lower and upper bounds \( l_i \) and \( \bar{l}_i (i = 1, 2, \ldots, n) \), respectively.

Step 2. Enter the option code (0 for a minimization problem and 1 for a maximization problem).

Step 3. Do the following for \( i = 1, 2, \ldots, n \).

(i) Divide the accepted region into two equal distinct subregions \( R_1 \) and \( R_2 \).

(ii) Applying RCGA, find the best found objective function value of \( f_1 \) and \( f_2 \) in the subregions \( R_1 \) and \( R_2 \), respectively.

(iii) If the option code is 0 (i.e. the problem is a minimization problem) and \( f_1 < f_2 \), accept region \( R_1 \); otherwise accept \( R_2 \). If the option code is 1 (i.e. the problem is a maximization problem) and \( f_1 < f_2 \), accept region \( R_2 \); otherwise accept \( R_1 \).

Step 4. Compute \( w_j = \bar{l}_j - l_j, j = 1, 2, \ldots, n \).

Step 5. If \( w_j < \varepsilon \), where \( \varepsilon \) is a pre-assigned very small positive number, for \( j = 1, 2, \ldots, n \), go to step 6; otherwise go to step 3.

Step 6. Print the values of the variables including the objective function values with negligible width.

Step 8. Stop.

The above algorithm is processed in association with the RCGA described below.

**Algorithm 2**

Step 1. Set \( t = 0 \) (\( t \) represents the current generation).

Step 2. Initialize \( P(t) \), the population at \( t \)th generation.

Step 3. Evaluate \( P(t) \).

Step 4. Find the best individual/chromosomes from \( P(t) \).

Step 5. Do the following until the termination condition is satisfied.

(i) Increase \( t \) by 1.
Global optimization for multi-modal functions

(ii) Select $P(t)$ from $P(t-1)$ by the ranking selection method.
(iii) Alter $P(t)$ by multiple parents, whole arithmetical cross-over, and non-uniform mutation with varying probabilities.
(iv) Evaluate $P(t)$.
(v) Find the best individual/chromosomes from $P(t)$.

Step 6. Print the best individual/chromosomes.
Step 7. Stop.

The following components are considered in the implementation of the above GA (algorithm 2) in algorithm 1:

- parameters of the GA
- initialization of the population
- evaluation
- genetic operators (selection, cross-over, and mutation).

4.1 Parameters of the genetic algorithm

The principal parameters of a GA are population size ($p\_size$), cross-over rate, mutation rate, and the maximum number of generations ($m\_gen$) or relevant stopping criteria. Population size is clearly problem dependent, and will need to increase with the dimensions of the problem. However, population size is restricted by both computing time and space limitations. As might be expected, GAs tend to be ineffective if very small populations are used, and so the choice of population size is always a compromise. Generally, rates of cross-over here range from 0.70 to 0.80, and the mutation rate varies from 0.15 to 0.20.

4.2 Initialization

The design of an appropriate chromosome representation of solutions of the problem is an important task in designing a GA. Several different representations can be used, of which binary coding and real coding are most popular. Classical binary coding is not suitable for the optimization of functions of continuous variables as in many cases it may not produce accurate solutions. Therefore we focus on the utilization of a real coding representation. In this representation, each chromosome is encoded in the form of an array of real numbers in which each component represents a variable of the function. To find the solution of a problem, initially a population (sequence of chromosomes) is created up to a desired number of chromosomes $V_1, V_2, V_3, \ldots, V_{\text{POPSIZE}}$, where each component is generated randomly within the bounds of the corresponding decision variables. This can be done in different ways. In our work, we have used a uniform distribution to find a random number within the specified bounds.

4.3 Evaluation

After obtaining a population of potential solutions to the problem, we have to calculate the fitness value of each chromosome. The fitness value plays an important role in GA and is obtained by considering different fitness functions. The fitness function is chosen in such a way that the highly fitted chromosomes (possible solution) have high fitness values. It is the only index used to select a chromosome to reproduce the next generation. We have chosen the objective function of the optimization problem as the fitness function.
4.4 Selection/reproduction operator

The primary objective of selection/reproduction is to emphasize good solutions and to eliminate bad solutions from the population, keeping the population size constant. This can be done by performing the following tasks.

(i) Identify good solutions (usually above average) in a population.
(ii) Make multiple copies of good solutions and eliminate bad solutions from the population so that multiple copies of good solutions can be placed in the population.

Common procedures/methods used to perform these tasks are ranking selection, roulette wheel selection, truncation selection, tournament selection, stochastic universal sampling, etc. We have used ranking selection, where the population is sorted from best to worst with selection probability assigned to each individual according to their ranking. The process is explained precisely in the algorithm given below.

**Selection algorithm**

Step 1. Sort all the fitnesses $f_i (i=1, 2, \ldots, p\text{-size})$ in decreasing order for maximization problems and increasing order for minimization problems and change their chromosomes accordingly.

Step 2. Generate a random number $c \in [0, 1]$.

Step 3. Calculate the probability $p_i$ of selection for each chromosome $V_i$ using the formula

$$p_i = c(1-c)^{i-1}.$$  

Step 4. Compute the cumulative probability $q_i$ for each chromosome $V_i$ using the formula

$$q_i = \sum_{j=1}^{i} p_j.$$  

Step 5. Generate another random real number $d$ in $[0,1]$.

Step 6. Obtain the minimum $k$ such that $q_k > d$ and select the $k$th individual/chromosome.

Step 7. Repeat step 6 until the number of selected individuals is equal to the population size.

4.5 Cross-over operator

This operator enhances the efficiency of the GA resulting in a more rapid approach towards the optimal space. This operator reshuffles the genetic material between two or more randomly selected parent chromosomes to produce improved offspring. We have used a multi-parent whole arithmetical cross-over with a variable probability rate which is a decreasing function of population age to perform the cross-over operation. Initially, it takes the highest prescribed value (say $p_c(0))$, and then decreases consecutively to obtain the lowest final value (say $p_c(m\_gen)$). Hence the variable probability rate $p_c(t)$ in the $t$th generation of population has the form

$$p_c(t) = p_c(0) \exp(-at)$$

where

$$\alpha = \frac{1}{m\_gen} \log \left( \frac{p_c(m\_gen)}{p_c(0)} \right).$$

Clearly, $0 < p_c(t) < 1$ as $0 < p_c(0) < 1$, and $0 < \exp(-at) < 1$, and it will be constant when $p_c(m\_gen) = p_c(0)$. 

The steps for the cross-over operation in the $t$th generation are as follows.

Step 1. Find the rounding-off integral value of the product of $p_c(t)$ and $p_{size}$ (say $N$).

Step 2. Generate a random number $r$ in $[0,1]$.

Step 3. Select randomly $N$ chromosomes for cross-over by universal stochastic sampling selection.

Step 4. Select three chromosomes from the $N$ chromosomes selected earlier and arrange them in descending order according to their fitness values.

Step 5. Produce two offspring, keeping the chromosome with higher fitness value the same. One of these two offspring will be generated by convex combination of the first two chromosomes the first parent having higher weighting. Similarly, the other offspring will be produced by convex combination of all three parent chromosomes with the first two parents having higher weighting. After the cross-over operation, the offspring will be as follows

$$
x_t^{*1} = x_t^1
$$

$$
x_t^{*2} = \lambda x_t^1 + (1 - \lambda)x_t^2
$$

$$
x_t^{*3} = \gamma x_t^1 + \delta x_t^2 + (1 - \gamma - \delta)x_t^3
$$

where $\lambda$, $\delta$, and $\gamma$ are real numbers in $(0,1)$ and $\lambda > 0.5$, $\gamma + \lambda < 1$, and $\gamma > \delta > 1 - \gamma - \delta$.

Step 6. Repeat steps 4 and 5 $N/3$ times (if $N$ is divisible by 3) or $N/3 + 1$ times otherwise.

Step 7. Stop.

To test the effect of the proposed cross-over, we have compared the average parent fitness with offspring (child) fitness. For this purpose, we computed the average fitness of the parent and the fitness of each offspring after a few thousand cross-over operations. Then we plotted the scatter graph of average parent fitness versus offspring fitness by normalizing the values for the second offspring only and we also computed the correlation coefficient of that offspring (figure 1). It can be seen that the points lie roughly along a line at an angle of 45° to the horizontal axis. The correlation coefficient between average parent fitness and fitness of the second offspring is +0.996881447 which is positive and close to unity. Therefore the proposed cross-over operation always gives improved offspring. This indicates that in every generation there is an improvement in the quality of the offspring. As this operation is performed several times, the offspring created at the end of each generation are highly energetic in the sense that they are moving towards the optimal point.

Figure 1. Scatter plot showing the correlation between average parent fitness and second offspring fitness.
4.6 Mutation operator

The mutation operator induces genetic diversity in the population by changing the value of a gene of the chromosome. It investigates the search space thoroughly from its local neighbourhood to capture the optimal solution. It is responsible for the fine tuning capability of the system and to regain information lost due to selection and cross-over in earlier generations. It is applied to a single chromosome and is performed with a low probability.

We have applied non-uniform mutation, with a variable rate of probability in the range \([p_m(m_{\text{gen}}), p_m(0)]\), where both \(p_m(m_{\text{gen}})\) and \(p_m(0)\) are prescribed. Initially, it takes a high value and then decreases consecutively as follows:

\[
p_m(t) = p_m(0) \exp(-\beta t)
\]

where

\[
\beta = \frac{1}{m_{\text{gen}}} \log \left( \frac{p_m(m_{\text{gen}})}{p_m(0)} \right)
\]

and \(t\) is the generation number. Clearly, the variable mutation rate is a proper fraction as \(0 < p_m(0) < 1\) and \(0 < \exp(-\beta t) < 1\), and it will be uniform when \(p_m(m_{\text{gen}}) = p_m(0)\).

This mutation operator depends on the age of the population. If the element (gene) \(V_{ik}\) of chromosome \(V_i\) is selected for this operation and the domain of \(V_{ik}\) is \([l_k, u_k]\) where \(l_k\) and \(u_k\) are the lower and upper bounds of the variable corresponding to the gene \(V_{ik}\), the reduced value of \(V_{ik}\) is given by

\[
V_{ik}' = \begin{cases} 
V_{ik} + \Delta(t, \hat{t}_k - V_{ik}), & \text{if the random digits is 0} \\
V_{ik} + \Delta(t, V_{ik} - \hat{l}_k), & \text{if the random digits is 1}
\end{cases}
\]

where \(k \in \{1, 2, 3, \ldots, n\}\) and \(\Delta(t, y)\) has a value in the range \([0, y]\).

In our experiment, we have used

\[
\Delta(t, y) = yr \left( 1 - \frac{t}{m_{\text{gen}}} \right)^b
\]

where \(r\) is a random real number in \([0,1]\), \(t\) is the current generation number, and \(b > 0\) is a constant (non-uniform mutation parameter).

5. Numerical examples

The algorithm was tested using the functions listed in Appendix A. These functions have different features such as continuous/discontinuous, convex/non-convex, unimodal/multi-modal, low dimension/high dimension. Each problem was solved in two phases: SGA was used in phase I and the proposed algorithm with RCGA was used in phase II. The following parameters were used for both SGA and RCGA in all problems, except where indicated otherwise.

- Phase I: \(p_{\text{size}} = 70; m_{\text{gen}} = 40\); constant probability of cross-over, 0.80; constant probability of mutation, 0.10.
- Phase II: \(p_{\text{size}} = 30; m_{\text{gen}} = 15\) (in problem 13, \(p_{\text{size}} = 50, m_{\text{gen}} = 30\)); \(p_c(0) = 0.80; p_c(m_{\text{gen}}) = 0.70; p_m(0) = 0.20; p_m(m_{\text{gen}}) = 0.15, \varepsilon = 1.0 \times 10^{-6}; \lambda = 0.8; y = 0.5; \delta = 0.3\).

Twenty-five independent runs were performed for each example in both phases by SGA and the proposed hybrid algorithm, in which the best found value has been accepted.
6. Sensitivity analyses

Sensitivity analyses were performed graphically and statistically to study the effect of changes of various parameters on the lower limit of the interval value number of the objective function for problems 1 and 13, as both these problems are of the minimization type. This experiment is performed only in phase II by changing one or more parameters at a time, keeping the other parameters at their original values. The results of these analyses are shown in figures 2–5 and tables 1–4, which are self-explanatory. In each case, 25 independent runs were performed for each test problem using the proposed algorithm. The lower limit of the interval-valued number of the objective function was considered in each run. To show the convergence of the algorithm graphically, the average values and the best objective functions were computed for different values of the parameter and plotted in figures 2–5. Tables 1–4 were constructed similarly, considering the best found value and the mean and standard deviation of the objective function for 25 sets of data obtained from 25 independent runs using the algorithm.

[Figures 2-4 showing the effect of changes of parameters on the objective function for different test problems]

Figure 2. Effect of changes of $p_{size}$ on the objective function of test problem 1 for $m_{gen} = 15$.

Figure 3. Effect of changes of $m_{gen}$ on objective function of test problem 1 for $p_{size} = 30$.

Figure 4. Effect of changes of $p_{size}$ on objective function of test problem 13 for $m_{gen} = 20$. 
Figure 5. Effect of changes of $m_{\text{gen}}$ on objective function of test problem 13 for $p_{\text{size}} = 50$.

Table 1. Effect of changes of $p_c(0)$ on the objective function.

<table>
<thead>
<tr>
<th>$p_c(0)$</th>
<th>Problem 1</th>
<th>Problem 13</th>
<th>Problem 1</th>
<th>Problem 13</th>
<th>Problem 1</th>
<th>Problem 13</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.85</td>
<td>$-1.8013039560$</td>
<td>0</td>
<td>$-1.8013039560$</td>
<td>$1.70 \times 10^{-9}$</td>
<td>0</td>
<td>$4.59 \times 10^{-8}$</td>
</tr>
<tr>
<td>0.83</td>
<td>$-1.8013039560$</td>
<td>0</td>
<td>$-1.8013039560$</td>
<td>$4.80 \times 10^{-9}$</td>
<td>0</td>
<td>$1.39 \times 10^{-8}$</td>
</tr>
<tr>
<td>0.80</td>
<td>$-1.8013039560$</td>
<td>0</td>
<td>$-1.8013039560$</td>
<td>$1.14 \times 10^{-8}$</td>
<td>0</td>
<td>$2.26 \times 10^{-8}$</td>
</tr>
<tr>
<td>0.72</td>
<td>$-1.8013039560$</td>
<td>0</td>
<td>$-1.8013039560$</td>
<td>$4.71 \times 10^{-8}$</td>
<td>0</td>
<td>$1.59 \times 10^{-7}$</td>
</tr>
<tr>
<td>0.70</td>
<td>$-1.8013039560$</td>
<td>0</td>
<td>$-1.8013039555$</td>
<td>$8.91 \times 10^{-8}$</td>
<td>4.00 $\times 10^{-10}$</td>
<td>$2.52 \times 10^{-7}$</td>
</tr>
</tbody>
</table>

Table 2. Effect of changes of $p_m(0)$ on the objective function.

<table>
<thead>
<tr>
<th>$p_m(0)$</th>
<th>Problem 1</th>
<th>Problem 13</th>
<th>Problem 1</th>
<th>Problem 13</th>
<th>Problem 1</th>
<th>Problem 13</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.20</td>
<td>$-1.8013039560$</td>
<td>0</td>
<td>$-1.8013039558$</td>
<td>$3.33 \times 10^{-10}$</td>
<td>4.00 $\times 10^{-10}$</td>
<td>$4.87 \times 10^{-10}$</td>
</tr>
<tr>
<td>0.19</td>
<td>$-1.8013039560$</td>
<td>0</td>
<td>$-1.8013039557$</td>
<td>$5.59 \times 10^{-9}$</td>
<td>5.00 $\times 10^{-10}$</td>
<td>$1.49 \times 10^{-9}$</td>
</tr>
<tr>
<td>0.18</td>
<td>$-1.8013039560$</td>
<td>0</td>
<td>$-1.8013039557$</td>
<td>$2.27 \times 10^{-8}$</td>
<td>5.00 $\times 10^{-10}$</td>
<td>$5.37 \times 10^{-8}$</td>
</tr>
<tr>
<td>0.17</td>
<td>$-1.8013039560$</td>
<td>0</td>
<td>$-1.8013039558$</td>
<td>$6.62 \times 10^{-8}$</td>
<td>4.00 $\times 10^{-10}$</td>
<td>$5.55 \times 10^{-7}$</td>
</tr>
<tr>
<td>0.16</td>
<td>$-1.8013039560$</td>
<td>0</td>
<td>$-1.8013039559$</td>
<td>$1.9 \times 10^{-7}$</td>
<td>3.00 $\times 10^{-10}$</td>
<td>$6.32 \times 10^{-7}$</td>
</tr>
</tbody>
</table>

Table 3. Effect of cross-over parameter $\lambda$ on the objective function.

<table>
<thead>
<tr>
<th>$\lambda$</th>
<th>Problem 1</th>
<th>Problem 13</th>
<th>Problem 1</th>
<th>Problem 13</th>
<th>Problem 1</th>
<th>Problem 13</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.55</td>
<td>$-1.8013039560$</td>
<td>0</td>
<td>$-1.8013039558$</td>
<td>$5.53 \times 10^{-9}$</td>
<td>4.00 $\times 10^{-10}$</td>
<td>$1.17 \times 10^{-8}$</td>
</tr>
<tr>
<td>0.60</td>
<td>$-1.8013039560$</td>
<td>0</td>
<td>$-1.8013039568$</td>
<td>$2.03 \times 10^{-8}$</td>
<td>4.00 $\times 10^{-10}$</td>
<td>$3.21 \times 10^{-8}$</td>
</tr>
<tr>
<td>0.65</td>
<td>$-1.8013039560$</td>
<td>0</td>
<td>$-1.8013039557$</td>
<td>$1.05 \times 10^{-8}$</td>
<td>5.00 $\times 10^{-10}$</td>
<td>$3.95 \times 10^{-8}$</td>
</tr>
<tr>
<td>0.70</td>
<td>$-1.8013039560$</td>
<td>0</td>
<td>$-1.8013039558$</td>
<td>$1.05 \times 10^{-8}$</td>
<td>4.00 $\times 10^{-10}$</td>
<td>$3.97 \times 10^{-8}$</td>
</tr>
<tr>
<td>0.80</td>
<td>$-1.8013039560$</td>
<td>0</td>
<td>$-1.8013039558$</td>
<td>0</td>
<td>4.00 $\times 10^{-10}$</td>
<td>$2.55 \times 10^{-6}$</td>
</tr>
</tbody>
</table>

Table 4. Effect of the cross-over parameters $\alpha$ and $\beta$ on the objective function.

<table>
<thead>
<tr>
<th>$\alpha$</th>
<th>$\beta$</th>
<th>Problem 1</th>
<th>Problem 13</th>
<th>Problem 1</th>
<th>Problem 13</th>
<th>Problem 1</th>
<th>Problem 13</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.500</td>
<td>0.470</td>
<td>$-1.8013039560$</td>
<td>0</td>
<td>$-1.8013039560$</td>
<td>$7.40 \times 10^{-8}$</td>
<td>0</td>
<td>$2.45 \times 10^{-7}$</td>
</tr>
<tr>
<td>0.525</td>
<td>0.430</td>
<td>$-1.8013039560$</td>
<td>0</td>
<td>$-1.8013039560$</td>
<td>$8.51 \times 10^{-7}$</td>
<td>0</td>
<td>$2.49 \times 10^{-5}$</td>
</tr>
<tr>
<td>0.550</td>
<td>0.390</td>
<td>$-1.8013039560$</td>
<td>0</td>
<td>$-1.8013039560$</td>
<td>$4.37 \times 10^{-8}$</td>
<td>0</td>
<td>$1.54 \times 10^{-7}$</td>
</tr>
<tr>
<td>0.575</td>
<td>0.340</td>
<td>$-1.8013039560$</td>
<td>0</td>
<td>$-1.8013039560$</td>
<td>$1.11 \times 10^{-8}$</td>
<td>0</td>
<td>$2.75 \times 10^{-8}$</td>
</tr>
<tr>
<td>0.600</td>
<td>0.310</td>
<td>$-1.8013039560$</td>
<td>0</td>
<td>$-1.8013039559$</td>
<td>$5.73 \times 10^{-8}$</td>
<td>3.00 $\times 10^{-10}$</td>
<td>$1.50 \times 10^{-7}$</td>
</tr>
</tbody>
</table>
7. Concluding remarks

A new hybrid method for solving multi-dimensional problems using an advanced GA and the interval technique has been proposed. This algorithm is hybrid in the sense that it utilizes the combined effect of the interval technique and the advanced GA and hence is different from any other method available in the literature. Initially, i.e. in phase I, the search space is reduced by dividing it into several subspaces which are handled in parallel by a simple GA. In phase II, the subspace (where the global optimum exists) obtained from phase I is reduced approximately to a point using the interval technique.

A new type of cross-over (multi-parent) has been successfully used for rapid convergence. In this operation, higher weighting is assigned to good individuals and lower weighting to bad and worse individuals. Thus the weighting parameters are taken as \( \lambda > 0.5, \gamma > \delta > 1 - \gamma - \delta, \text{ and } \gamma + \delta < 1. \)

The proposed method is applicable when the search space is very large and the objective function is multi-dimensional and multi-modal. In particular, this method is suitable for solving decision-making problems with uncertain search spaces. In this case, this method can be implemented by considering a large search space. It can be applied to a variety of engineering problems, such as image processing, robotics, and circuit design.

Acknowledgements

The authors would like to acknowledge support for this research provided by the University Grants Commission, India, under DRS phase III programme. The authors are also grateful to the anonymous referees for their constructive and helpful comments and suggestions.

Appendix A: Test functions

1. Michalewicz function (MZ) (n variables)

\[
MZ = -\sum_{i=1}^{n} \sin(x_i)\sin(i.(x_i^2/\pi))^{2m}, \quad m = 10, \quad n = 2
\]

Search domain: \(-\pi \leq x_i \leq \pi, \quad j = 1, 2, \ldots, n\)

One global minimum: \((x_1, x_2)^*\)

\(x_1 \in (2.202904, 2.202905), \quad x_2 \in (1.570794, 1.570795)\)

\[MZ = [(x_1, x_2)^*] = -1.801303956.\]

2. De Jong function F1(three variables)

\[
F1(x_1, x_2, x_3) = \sum_{i=1}^{3} x_i^2
\]

Search domain: \(-5.12 \leq x_i \leq 5.12, \quad i = 1, 2, 3\)

One global minimum: \((x_1, x_2, x_3)^*\)

\(x_1 \in (0.000000, 0.000000), \quad x_2 \in (0.000000, 0.000000), \quad x_3 \in (0.000000, 0.000000)\)

\[F1[(x_1, x_2, x_3)^*] = 0.\]
3. De Jong function F2 (two variables)

$$F2(x_1, x_2) = 100(x_1^2 - x_2)^2 + (1 - x_1)^2,$$

Search domain: $-2.048 \leq x_i \leq 2.048, i = 1, 2$
One global minimum: $(x_1, x_2)^*$
$$x_1 \in (0.999999, 0.999999), x_2 \in (0.999999, 0.999999)$$
$$F2[(x_1, x_2)^*] = 0.$$

4. Schaffer function F6 (two variables)

$$F6(x_1, x_2) = 0.5 + \frac{\sin^2 \sqrt{x_1^2 + x_2^2} - 0.5}{[1.0 + 0.001(x_1^2 + x_2^2)]^2}$$

Search domain: $-100 \leq x_i \leq 100, i = 1, 2$
One global minimum: $(x_1, x_2)^*$
$$x_1 \in (0.000000, 0.000000), x_2 \in (0.000000, 0.000000)$$
$$F6[(x_1, x_2)^*] = 0.$$

5. Schaffer function F7 (two variables)

$$F7((x_1, x_2) = (x_1^2 + x_2^2)^0.025[\sin^2(50(x_1^2 + x_2^2)^0.1) + 1.0]$$

Search domain: $-100 \leq x_i \leq 100, i = 1, 2$
One global minimum: $(x_1, x_2)^*$
$$x_1 \in (0.000000, 0.000000), x_2 \in (0.000000, 0.000000)$$
$$F7[(x_1, x_2)^*] = 0.000641.$$

6. Goldstein and Price (GP) function (two variables)

$$GP(x_1, x_2) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)]$$
$$\times [30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$$

Search domain: $-2 \leq x_i \leq 2, i = 1, 2$
One global minimum: $(x_1, x_2)^*$
$$x_1 \in (0.000000, 0.000000), x_2 \in (1.000000, -0.999999)$$
$$GP[(x_1, x_2)^*] = 3.000000.$$

7. Branin RCOS function (two variables)

$$RCOS(x_1, x_2) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10 \left(1 - \frac{1}{8\pi}\right)\cos(x_1) + 10$$

Search domain: $-5 \leq x_1 \leq 10, 0 \leq x_2 \leq 15$
Three global minima: $(x_1, x_2)^*$
Case 1: \(x_1 \in (-3.141593, 3.141592), x_2 \in (12.25005, 12.25006)\)
Case 2: \(x_1 \in (9.424778, 9.424778), x_2 \in (2.250007, 2.250008)\)
Case 3: \(x_1 \in (3.141593, 3.141593), x_2 \in (2.249993, 2.249994)\)

\[RCOS[(x_1, x_2)^*] = 0.397887694.\]

8. Six-hump camel-back function (two variables)

\[F(x_1, x_2) = \left(4 - 2.1x_1^2 + \frac{x_1^4}{3}\right)x_1^2 + x_1x_2 + (-4 + 4x_2^2)x_2^2\]

Search domain: \(-3 \leq x_1 \leq 3, -2 \leq x_2 \leq 2\)

Two global minima: \((x_1, x_2)^*\) at two different points
Case 1: \(x_1 \in (-0.089842, -0.089841), x_2 \in (0.712656, 0.712656)\)
Case 2: \(x_1 \in (-0.089842, 0.089843), x_2 \in (-0.712656, -0.712656)\)

\[F[(x_1, x_2)^*] = -1.031628.\]

9. Easom function (ES) (two variables)

\[ES(X_1, X_2) = -\cos(x_1)\cos(x_2)\exp\{-[(x_1 - \pi)^2 + (x_2 - \pi)^2]\}\]

Search domain: \(-100 \leq x_i \leq 100, i = 1, 2\)

One global minimum: \((x_1, x_2)^*\)
\(x_1 \in (3.141464, 3.141465), x_2 \in (3.141464, 3.141464)\)

\[ESI[(x_1, x_2)^*] = -1.000000.\]

10. Bohachevsky function-1 (BH1) (two variables)

\[BH1(x_1, x_2) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1) - 0.4\cos(4\pi x_2) + 0.7\]

Search domain: \(-50 \leq x_i \leq 50, i = 1, 2\)

One global minimum: \((x_1, x_2)^*\)
\(x_1 \in (0.000000, 0.000001), x_2 \in (0.000000, 0.000000)\)

\[BH1[(x_1, x_2)^*] = 0.\]

11. Bohachevsky function 2 (BH2) (two variables)

\[BH2(x_1, x_2) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1)\cos(4\pi x_2) + 0.3\]

Search domain: \(-50 \leq x_i \leq 50, i = 1, 2\)

One global minimum: \((x_1, x_2)^*\)
\(x_1 \in (0.000000, 0.000001), x_2 \in (0.000000, 0.000001)\)

\[BH2[(x_1, x_2)^*] = 0.\]

12. Bohachevsky function 3 (BH3) (two variables)

\[BH3(x_1, x_2) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1 + 4\pi x_2) + 0.3\]

Search domain: \(-50 \leq x_i \leq 50, i = 1, 2\)

One global minimum: \((x_1, x_2)^*\)
866  S. D. Jabeen and A. K. Bhunia

\[ BH^3[(x_1, x_2)^*] = 0. \]

13. Colville function \( F \) (four variables)

\[
F(x_1, x_2, x_3, x_4) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2 + 90(x_4 - x_3^2)^2 + (1 - x_3)^2
+ 10.1((x_2 - 1)^2 + (x_4 - 1)^2) + 19.8(x_2 - 1)(x_4 - 1),
\]

Search domain: \(-10 \leq x_i \leq 10, i = 1, 2, 3, 4\)

One global minimum: \((x_1, x_2, x_3, x_4)^*\)

\[ x_1 \in (0.000000, 0.000001), \quad x_2 \in (0.000000, 0.000001) \]

14. Shubert (SH) (two variables)

\[
SH(x_1, x_2) = \sum_{j=1}^{5} j \times \cos[(j + 1) \times x_1 + j] \times \sum_{j=1}^{5} j \times \cos[(j + 1) \times x_2 + j] \]

Search domain: \(-10 \leq x_j \leq 10, j = 1, 2\)

Eighteen global minima:

\[ SH[(x_1, x_2)^*] = -186.73090883. \]

References


