Multidimensional Mutation Evolutionary Algorithm

CORINA ROTAR
Computer Science Department
“1 Decembrie 1918” University
Alba Iulia, 11-13 N.Iorga street
ROMANIA
crotar@uab.ro

Abstract: The behavior of standard evolutionary algorithm in the case of multi-modal optimization problems meets a major difficulty. It generally converges towards a single optimum point failing to maintain in the population the multiple optima of the problem under consideration. Various methods enrich the standard algorithm to obtain efficient techniques for solving multi-modal problems. These methods mainly consist of increasing the population diversity and of maintaining the promising areas in the search space in order to finally achieve convergence of the population towards the multiple optima. The present paper introduces mmEA, an evolutionary algorithm for multimodal optimization based on multidimensional exploration of the search space. This technique doesn’t require any user defined parameter except those specific to standard evolutionary algorithm. Experiments and comparisons with similar techniques from literature, for static and dynamic environment, prove that mmEA technique is promising.

Keyword: multimodal optimization, multidimensional mutation.

1 Background
Mostly, finding only the global optimum is not sufficient in the problem context. Real world problems frequently have more than one optimum. Therefore, evolutionary techniques have been developed to offer multiple optima of the considered problem. In multimodal context, diversity of the population represents an important issue of the development of an efficient evolutionary algorithm. Standard evolutionary algorithm has an important drawback: the lack of population diversity and the convergence towards one optimum causes an impossibility of maintaining multiple optima of the problem. The research is focused on this direction of finding skillful evolutionary techniques for solving multimodal problems. Several significant improvements were made by adding extra mechanisms to evolutionary algorithms to deal with multimodality: the niching techniques inspired by the niche concept are suitable for maintaining several subpopulations (niches) corresponding to each optimum and these techniques are responsible for maintaining the genetic drift.

Crowding
De Jong [3] proposed crowding techniques, which are responsible for diversity preservation and, consequently, for maintaining multiple optima into the population. The main idea of these techniques is to provoke the individuals to establish around each optimum by using a replacement procedure based on the similarity degree. Standard crowding described by de Jong [3] and Deterministic Crowding [6] are two variants of the same approach.

A crowding technique is also used in CrowdingDE algorithm[10]. Basically, CrowdingDE is a Differential Evolution algorithm [9], supplied with a crowding mechanism for maintaining multiple solutions in the population. This algorithm is comprehensible and provides good results in multimodal optimization. A major advantage of the CrowdingDE is that it does not use additional user specified parameters (e.g. niche radius) to define the niches corresponding to multiple optima.

Sharing
Fitness sharing described in [4] represents a point of reference for multimodal optimization. In order to make the population converge towards multiple optima, the standard evolutionary algorithm is enriched by an extra mechanism in which case the population consists in several subpopulations corresponding to all promising regions of the search space. This partitioning of the population is suggested by the biological concept of ecological niches. Each subpopulation tends to occupy a specific zone, and therefore to converge to one optimum. The principle of sharing technique is to reduce the fitness of each individual of the current population by an amount proportional to the number of similar individuals of the population. For establishing the sharing degree among the individuals of the population a distance function \( d \) is chosen. This function could be Euclidian distance, Hamming distance, Levenstein distance. The larger the distance, the less the sharing degree.

Sharing technique uses a parameter called niche radius which has the meaning of the maximum allowed distance between two individual which they are
considered from the same niche. Setting proper value for the niche radius is crucial and requires a previous knowledge of distances among the optima. Generally these information are not available at the time of algorithm design, therefore, choosing the value for the niche radius is problematical.

Speciation

Speciation techniques ([0],[5],[8]) , inspired by a natural process of evolution, are also niching methods which exploit the concept of species. One of these techniques, SCGA [5] represents an evolutionary algorithm which makes use of species concept. The algorithm is based on the concept of partitioning the population into several subpopulations according to similarity degree of the individuals. Each of these subpopulations is called species and is built around a dominating individual called the species seed. All species seeds of the current population are conserved by moving them into the next generation.

A species is defined as a “class of individuals with the same characteristics” [5]. A supplementary parameter species distance indicates the maximum value of the distance between two individuals for which they are considered similar. Therefore, a species is a subpopulation for which the distance between any two individuals is less than the species distance. The species distance is also used to decide which individuals should be preserved from one generation to the next one.

Setting the proper value for the species distance represents an important disadvantage of the technique.

Clearing

The clearing technique [7] is based on the niching principle according to which the subpopulations share the limited resources of the same area. There are similarities with the sharing techniques, but in clearing procedure’s case, each subpopulation has a winner (the individual with the best fitness) which will fully gain the whole resources, while the others’ fitness will be set to zero. The procedure can be generalized by accepting more than one winner on each niche.

In order to establish the niches, a supplementary parameter is used: clearing radius. The significance of this parameter is equivalent to the niche radius defined in sharing procedures.

2 Description of mmEA

The paper introduces an algorithm named Multidimensional Mutation EA aimed for solving optimization problem in static and dynamic multimodal environment. Without using an additional mechanism to increase the population diversity, using only an adaptive multidimensional mutation, the algorithm proves to be efficient in the test problems under consideration.

The particular nature of the proposed algorithm resides in the scheme of the mutation operator. The algorithm induces subpopulations that are able to settle in the multiple optima areas of the problem under consideration. Moreover, in a dynamic environment, without additional procedure, the proposed technique is able to quickly adapt the population to the changeable landscape.

The aim is to design a general and simple evolutionary technique able to solve optimization problems in static environment and in dynamic environment without any supplementary mechanisms or important transformation of its structure.

2.1 Multidimensional mutation

The specificity of the algorithm we describe in this paper is the proposed mutation operator. There are no recombination or selection mechanisms involved. The only search operator is the multidimensional mutation.

Considering the following single criteria optimization problem:

\[ f : D \to R, \quad D = [a_1, b_1] \times [a_2, b_2] \times \ldots \times [a_n, b_n] \subseteq \mathbb{R}^n \]

\[ f(x) = \max \{ f(x), \quad x \in D \} \]

Each individual of the current population is represented by a vector:

\[ c = (x_1, x_2, \ldots, x_n), \quad \forall i \in \{1, 2, \ldots, n\}, \quad x_i \in [a_i, b_i]. \]

An individual is identified by its position into the population. We will refer the \( i \)-th individual unconstrained by the current generation number or the genetic structure of it. Therefore, the genetic modification is interpreted as an evolution of the considered individual, not as obtaining the new offspring.

Evaluation of each individual is made by involved objective function. Also, an additional value is assign to each individual. This value represents the mutation step of the individual for the next stage of evolution. Along the evolution of the population, we compute the average improvement of every individual: the average growth is given by the average value of the differences between the current performance of the individual and its previous performance in the last generations.

\[ \text{growth}(c, t) = \frac{1}{t} \sum_{k=1}^{t} \text{fitness}(c_{i}^{'}) - \text{fitness}(c_{i}) \quad (4) \]

Where:

- \( c_k \) - individual at generation \( k \)
- \( c_{i}^{'} \) - mutated individual at generation \( k \)
- \( t \) – current generation
The current growth is calculated by difference between the current performance of individual and its recorded performance at previous generation:

\[ \text{current\_growth}(c, t) = \text{fitness}(c') - \text{fitness}(c) \] (5)

If the current growth is larger than the average growth of an individual \( c \) at generation \( t \), we consider that the individual suffers an appropriate adaptation, thus, further modifications of this individual will be smaller. By this mechanism of changing the mutation step, the loss of the potential optimal solutions from the nearby vicinity of that individual is prevented. The mutation step is computed accordingly to the adaptation coefficient of the specific individual. The adaptation coefficient is given by the next formula:

\[ AQ(c) = \begin{cases} AQ(c) + p(c,t), & \text{if current\_growth}(c, t) > \text{growth}(c, t) \\ AQ(c), & \text{otherwise} \end{cases} \] (6)

Where: \( p(c,t) \) – represents the penalization value for individual \( c \) at generation \( t \). Each time the performance of an individual gets bigger, its adaptation coefficient is increased and the next step of mutation becomes smaller in order to encourage the further fine-grained movements in the specific region. The penalization value is dynamically computed accordingly to the growth of the specific individual along its evolution:

\[ p(c, t) = \text{growth}(c, t) \] (7)

The mutation step is given by the following formula:

\[ \text{Step}(c) = \frac{1}{1 + AQ(c)} \] (8)

If an individual increases its performance, the further mutations which it suffers will be made by smaller steps. Initial values of \( AQ \) coefficient are 0 for the entire population.

By mutation, each individual tries to find the best new position in the search space. Thus, each gene can be changed by adding or subtracting a value corresponding to the mutation step. The total number of new positions is \( n \), where \( n \) is the length of the chromosome and the dimension of the search space. The best new found position is encoded in chromosome \( c' \). The improved chromosome \( c' \) is next considered for replacement. The survival mechanism consists in replacing \( c \) by \( c' \) only if performance of \( c' \) is better.

**Multidimensional Mutation**

For each individual \( c \) of population \( P \) execute

1. The individual \( c \) is cloned in \( n \) instances \( \{c_1, c_2, ..., c_n\} \).
2. For each \( j \) from 1 to \( n \)
   The \( j^{th} \) clone of the parent will be changed as follows:
   2.1 Generate random value \( v' \) from \([0, \text{Step}(c)]\).
   2.2 Adjust the \( j^{th} \) gene as follows
      Randomly generate \( q \in [0,1] \)

\[\begin{align*}
\text{If} (q < 0.5) & \quad x_i^{*} = x_i - (x_i - a_i) \cdot v' \\
\text{Else} & \quad x_i^{*} = x_i + (b_i - x_i) \cdot v'
\end{align*}\]

If \((q < 0.5)\) then \( x_i^{*} = x_i - (x_i - a_i) \cdot v' \)
Else \( x_i^{*} = x_i + (b_i - x_i) \cdot v' \)

**Multidimensional Mutation Evolutionary Algorithm**

The method described above is simple and intuitive. We consider an objective function defined in the plane, with real values:

\[ f : D \rightarrow \mathbb{R}, \quad D \subseteq \mathbb{R}^2 \]

Its graphic representation is given by a surface. We assume that we need to determine the maxima on the definition domain \( D \). If initially, the individuals of the population are placed randomly on the surface described by the function \( f \), the evolutionary algorithm forces the population to climb by a rather small step to the surface peaks and find the multiple maxima of the function under consideration.

For a good choice of the mutation step and sufficiently high value of parameter \( N \) – number of individuals, the population tends to occupy all the high areas of the surface representing the promising areas of the search space.

The adequate size of the “forwarding” step compels the individuals to head towards the closest peak in the occupied area. The mutation step is adapted accordingly to the improvement capacity of the individual. If an individual recorded an increased performance, its further mutation step will be decreased in order to efficiently search the corresponding local zone. Otherwise, the mutation step value remains unaffected.

The initial uniform position of the chromosomes in the search space and their sufficiently large number influence the detection of all optimum points.

A shortcoming of the method is that the number of the found optima depends on the size of the population and on the initial population. It has been found that for a too small size of the population and for an unbalanced position of the individuals in the initial population, certain promising areas of the space are ignored without the possibility to shift to them later. The algorithm can overcome this flaw if it is endowed with a proper additional mechanism which could determine the return or migration of the population in the unexplored areas. Likewise, using a larger population would have the same effect.

We have chosen the second solution in our present research as it is easier to implement, although this leads to higher calculation costs.

**mm -Evolutionary Algorithm**

1. Randomly generate the initial population \( P(0) \), \( t=0 \).
2. Evaluate individuals of \( P(0) \)
3. While (stopping\_condition=false) execute:
For each \(i^{th}\) individual of population \(P(t)\) do

1. Find the next position, encoded by \(c'\)
2. Evaluate \(c'\)
3. \(c'\) replaces the parent if it is better than the parent
4. Compute the AQ coefficient of new \(i^{th}\) individual of the population
   \[ t:=t+1 \]

### 3 Experimental results

In order to validate our proposal we conduct numerical experiments on well-known test functions, comparing the mmEA’s results with the result obtained by other popular algorithms for multimodal optimization: CrowdingDE, SCGA, and Genetic Algorithm with Clearing. Each of the mentioned algorithms use one of the niching methods described in introductory part of the paper. Due to the lack of space we shall describe next only the results obtained for two designed test functions for multimodal optimization, known as Rastrigin and Schubert functions.

**Setting the specific parameters for the algorithms:**

Excepting mmEA, the other algorithms are parameter dependent.

**SCGA’s specific parameters:** The species distance has a very important role. Setting an inappropriate value for the species distance decreases the efficiency of the technique. In [2] it is proposed that the species distance be estimated as follows. The radius of the smallest hypersphere containing feasible space is computed by:

\[
 r = \frac{1}{2} \sqrt{\sum_{i=1}^{n} (b_i - a_i)^2} \tag{9}
\]

where \(a_i\) and \(b_i\) are the upper and lower bound of the \(i^{th}\) control variable.

Therefore, if the number of global optima \(N_g\) is known, the species distance can be estimated as:

\[
 \sigma_s = \frac{r}{\sqrt[N_g]{N}} \tag{10}
\]

In our experiments we use this procedure of estimating the species distance, as the authors of SCGA, for all test functions, excepting the Shubert function. The authors of SCGA reported that the best performance for Shubert test function is given by the specific value 1.6 for the species distance. Therefore, we choose the same value of the species distance to assure a correct comparison in our experiments.

**Clearing Algorithm’s parameters:** The clearing technique has an additional parameter, clearing radius, which is very important in developing the suitable algorithm. The purpose of clearing radius is analogous to the role of niche radius in sharing approaches or to the species distance in speciation techniques. The same disadvantage is induced by the need of setting the proper value for clearing radius. The result of the clearing algorithm is very much influenced by the specific user defined parameter. The formula (10) can be easily adopted for estimating the right value for clearing radius. In our experiments, according to the above mentioned formula, we use the same values for the similar parameters of the involved algorithms: species distance and clearing radius.

**CrowdingDE’s parameters:** CrowdingDE does not use any parameter with the same significance of the niche radius. Even CrowdingDE separates the population in niches, the process is different and the replacement procedure is responsible for that effect. Therefore, there is no extra parameter to estimate the size of a niche and in consequence, the algorithm’s results are hardly affected by user defined parameters.

**Other parameters:** regular parameters of evolutionary algorithms: size of the population and number of produced generations are set to the following values, for all considered algorithms: population size \(\text{DimPop}=100\), maximum number of generation \(T_{\text{max}}=100\).

Each algorithm runs for 30 times. The population size is set to 100 individuals and the maximum number of produced generations is 100. Regarding the specific parameters for each algorithm (species distance, clearing radius), we follow the suggestion found in [2] and described before in this paper. The results obtained by each run of each compared algorithm are analyzed. mmEA proves to be more effective in finding multiple optima of the considered test functions. In most of the cases, the results obtained by mmEA overcome the results obtained by other algorithms.

**Rastrigin function** is a highly multimodal test function. The goal is to find all minima (local and global). For two decision variable and for the considered search domain, Rastrigin function has 121 minima equally distributed. All four compared algorithm prove to be able to find the global minimum and also several local minima closely positioned to the global optimum. Regarding the number of optima found, mmEA and CrowdingDE overcome SCGA and Clearing Algorithm. Knowing the exact location of optima, we compute the number of found optima in the last generations by using a precision value: \(\varepsilon=0.1\).

Setting the right value for the species distance and clearing radius is problematic in these experiments; we use the Deb&Goldberg’s formula (10) with two different values for the expected optima (Table 3).

A smaller value for clearing radius offers better results. The best results are further compared with the results obtained by mmEA, CrowdingDE.
mmEA and CrowdingDE prove a very good behavior on Rastrigin test function. For a population of 100 individuals, an average number of 90 individuals are good approximations of the real optima. SCGA demonstrates to perform better that Clearing Algorithm, finding more optima with the specific accuracy.

For an objective comparison we conducted the following experiments: each algorithm runs for 30 times, with 100 individuals and 100 generations. For Shubert function we know that \(x^*\) is an optimum if \(f(x^*) = -186.7309\). Using this information, in order to compute de number of found optima we consider that an individual of the last generation approximates an optima if the absolute difference between its function value and the optimum function value \((-186.7309)\) is less than \(\varepsilon\). We vary this threshold, \(\varepsilon\), to analyze the accuracy of the results.

We note that mmEA finds multiple optima with highest accuracy among all other algorithms. Only for a larger value \(\varepsilon = 10\), CrowdingDE’s performance gets closer to mmEA’s performance. SCGA outperforms Clearing for the considered parameter.

As for the Shubert test function, a valuable investigation is reported in [5]; We will focus on comparing our algorithm with the SCGA. The authors states that SCGA finds all 18 solutions of Shubert function in 100 generations, using a large population of 1000 individuals. For such a large size population, SCGA performs an average number of function evaluations of 64178, in an average of 89 generations for detecting all 18 solutions.

From the results of SCGA reported elsewhere, the lowest value for the average function evaluations is recorded for a population of 200 individuals in an average value of 250 generations. The average number of function evaluation is reported as 35647. However, there is too little information provided regarding the accuracy of the solutions. A solution \(x\) is considered found when the following condition is verified:
\[ f(x) = (f_{\text{max}} - f_{\text{min}})r_f \]

where: \( f_{\text{max}}, f_{\text{min}} \) represents the maximum and the minimum fitness value of the current population, and parameter \( r_f \) is 0.95. We exploited the same condition to identify the optima. The stopping condition of mmEA is given by the determination of all optima. Therefore, for a population of 100 individuals, in 30 runs, mmEA provides all 18 optima in an average number of only 81 generations, with an average number of 32360 function evaluations. This result allows us to consider mmEA at least of an equivalent efficiency as its competitor regarding the computational cost. Moreover, the experimental results summarized in Table 3 suggest that mmEA offers more accurate solutions than any other algorithm.

4 Conclusions

In this paper, we presented mmEA, a comprehensible algorithm for solving multimodal optimization problems. The technique is based on using a special mutation procedure combined with an elitist replacement of each parent individual with the best mutant.

The main advantages of mmEA consist in its independence by the user defined parameters, the natural utilization in dynamic environment without any adjustment, and better results than other popular algorithms for different test functions. Our experimental results show that for all considered test functions in static environment mmEA wins the competition with other three algorithms corresponding to popular techniques to deal with multimodality: speciation, clearing and crowding. Moreover, mmEA provides good results without previous knowledge of the optima position; SCGA and Clearing Algorithms require establishing the right value for an equivalent parameter (species distance, clearing radius).

The proposed algorithm, mmEA, does not use the replacement strategy as CrowdingDE. The descendant substitutes its parent if an improvement occurs. Its good behavior is provided by the adaptation of mutation step. The principle is simple: an individual comprises its achievement during evolution. An individual’s achievement is represented by the average of the differences between its current performance and its previous performance during the last generations. When an individual records a significant improvement, it means generally that it is located near an optimum, and so, its further mutations become smaller to facilitate an efficient local search. Otherwise, if an individual’s performance along its evolution does not increase substantially, it generally means that the specific individual did not get closed to an optimum and thus, the mutation step remains unchanged to assure the global search of the search domain.

The technique described in this paper proves to overcome CrowdingDE in static environment. mmEA is able to find multiple. With few irrelevant exceptions, the results of mmEA overcome CrowdingDE’s results regarding the number of found optima and the optima’s accuracy, in all test scenarios. Another advantage of mmEA consists in its comprehensibility and easy implementation. Moreover, the algorithm can be simply applied for dynamic optimization problems without any additional mechanism to assure the population’s adaptation to the possible changes in the environment.

References: