Dynamic Multimodal Function Optimization using Genetic Algorithms

Walter Cedeño
Computing Research Group/CMRD
Lawrence Livermore National Laboratory
Livermore, CA 94550 (wcedeno@llnl.gov)
and
Venkateswararao Vemuri
Department of Applied Science and LLNL
University of California, Davis
Livermore, CA 94550 (vemuri@icdc.llnl.gov)

Abstract

Many optimization techniques work well for unimodal functions. If applied to multimodal functions, they tend to converge to only one of the many peaks. Optimization of multimodal functions becomes even more difficult if the function parameters change dynamically. Genetic algorithms have been successfully applied by several investigators for static optimization of multimodal functions. This modest success is primarily due to the ability of genetic algorithms to locate more than one peak. In this paper we introduce a combination of selection and replacement operators that is suitable for multimodal function optimization in a dynamic environment using various test functions, performance of this new operator is studied. Utility of this new operator to multimodal function optimization in a dynamic environment is described.

Keywords: genetic algorithms, multimodal functions, dynamic optimization.

1. Introduction

The problem of finding the maxima of functions with multiple peaks is difficult to solve using traditional optimization methods. Moreover, in some problems we want to find not only the highest peak, but all the other peaks as well. The problem becomes harder when the parameters of the function being maximized are changing dynamically. Problems of this type can be found in airline traffic control systems, satellite scheduling, and transportation scheduling and routing [1]. Good solutions are hard to find due to the problem complexity and good techniques, are most of the time, problem dependent. As the size of the problem becomes larger current techniques often produce solutions far from optimum at a very high computational expense. In this paper we describe a selection operator called *crowding selection* and a replacement operator applicable to dynamic multimodal function optimization. Our goal is to construct a GA model that is appropriate for this type of function optimization. Two cases are considered: static (the function parameters remain fixed), and dynamic (the parameters are allowed to vary).

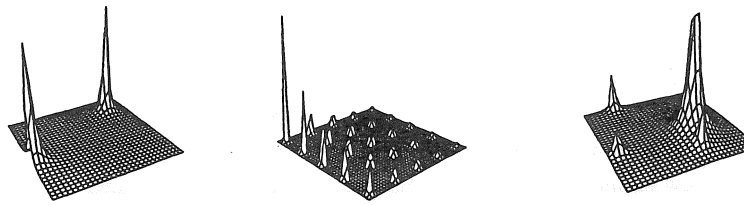
Genetic algorithms were introduced by Holland [9] in the 1970s, and have generated widespread interest in the last ten years. GAs are general purpose search procedures based on the principle of natural selection and genetic recombination. As in nature, GAs use the mechanics of evolution to improve a set of initial solutions using "recombination" and "mutation" of the "genetic material". This process ensures that good solutions are selected more often for recombination and generation of new solutions. During recombination new solutions are created and inserted into the current family of solutions; old and ineffective solutions are allowed to perish. Selection of mating partners, mating process, and replacement techniques are driven by genetic operators.

Genetic Algorithms differ from traditional techniques in several ways. GAs work with several solutions at the same time and improve them by recombining their good features while exploring new solutions in the search space. This approach has the advantage of permitting good solutions to guide the algorithm to near optimal solutions and at the same time prevents from getting stuck at local extrema. The genetic operators are governed by probability and are problem independent; the method can easily be applied to many problems. GAs use binary strings, to represent the problem, which are easy to manipulate thus facilitating fast execution of the programs.

Section 2 of this paper describes the set of test functions utilized to test different genetic operators, and benchmarks used to measure their performance. In section 3, we will introduce the GA model appropriate for multimodal function optimization. Section 4 presents the results obtained on the test functions and section 5 contains a summary and conclusions.

2. Function Definitions

To evaluate the performance of our GA model we use four different multimodal functions F1 to F4. The functions were chosen for their diversity: all peaks with approximately the same height, peaks with widely varying heights, and dynamically changing peaks. Figure 1 shows examples of these functions. Function F1 contains two peaks of the same height and width located far apart. Function F2 contains 25 peaks of different heights located in a 5x5 square region, the minimum height peak having been located in one corner and the maximum height peak in the opposite corner. This is one of the De Jong test function [4]. Due to its symmetry it is not sufficient to use this function to test the GA performance on multiple height peaks. Function F3 contains multiple peaks generated at random. The location, height, and width of each peak are all determined at random at runtime. Function F4, which is not shown, is made out of the sum of two different five-peak functions like F3; the contribution of each function is changed dynamically by multiplying it by a constant varying between 0.0 to 1.0. Different rates of contribution varying from 1% every generation to 1% every 10 generations were tried.



(a) (b) (c)

Figure 1: (a) Function F1 with similar peaks.
(b) Function F2 with multiple dissimilar peaks.
(c) Function F3 with randomly located peaks.

With the exception of F2, each peak in a function is generated by the formula:

$$F(x, y) = \frac{height}{1 + width * ((x - XCenter)^2 + (y - YCenter)^2)},$$

where *height* determines the maximum value for the peak, *width* the base width of the peak, and *XCenter* and *YCenter* the location of the peak in the x-y coordinate system. F2 is generated using the formula:

$$F(x, y) = 0.002 + \sum_{i=1}^{25} \frac{1.0}{i + (x - A_{1i})^6 + (y + A_{2i})^6},$$

where A_{1i} and A_{2i} respectively are the x and y coordinates of the i-th peak.

$$A = \begin{bmatrix} -32 & -16 & 0 & 16 & 32 & -32 & -16 & 0 & 16 & 32 & -32 & -16 & 0 & 16 & 32 & -32 & -16 & 0 & 16 & 32 & -32 & -16 & 0 & 16 & 32 \\ -32 & -32 & -32 & -32 & -32 & -16 & -16 & -16 & -16 & -16 & 0 & 0 & 0 & 0 & 0 & 16 & 16 & 16 & 16 & 16 & 32 & 32 & 32 & 32 & 32 \end{bmatrix}.$$

F2 [4] was chosen as a test function to compare the performance of our GA model in a search space where the taller peaks are situated near a corner. It also provides a complex search space where we can examine how the distribution of solutions are affected as the algorithm is run for longer periods.

3. Genetic Algorithms and Operators

Central to GAs are the twin concepts of natural selection and gene recombination. These algorithms manipulate strings of bits called *chromosomes,* which encode values in the search space. With this simple encoding, the genetic algorithm evolves a set of solutions, called a *population* , using the same mechanisms found in nature. They operate without any knowledge of the decoded search space. The only information they have is the *fitness* value (score) for each *organism* (solution) in the population and use it only to bias the selection of organisms for mating and/or replacement. Those organisms with higher fitness will tend to reproduce more and replace less fit organisms from the population.

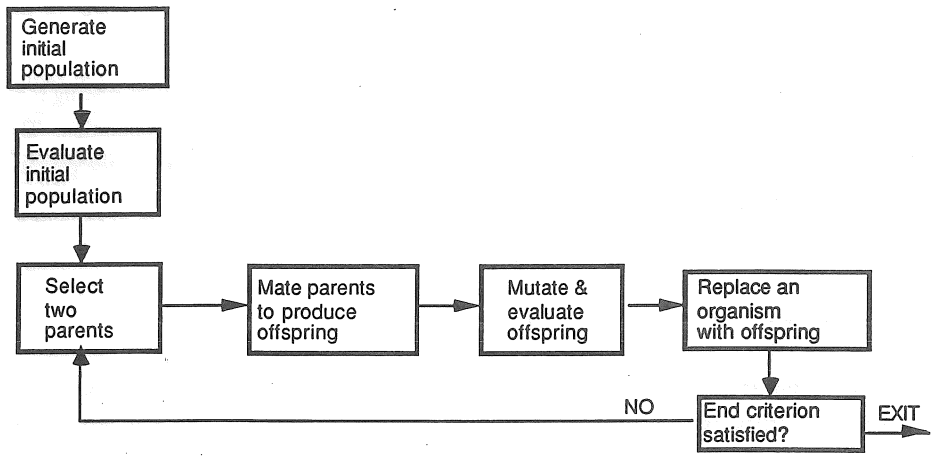


Figure 2: Basic structure of a genetic algorithm based on a steady-state model

Most of the material describing GAs uses the so-called *generational model*. In this model a new population replaces the current population entirely. This model lacks biological realism because organisms may survive various generations in a natural setting. In this paper we use what is known as a *steady-state* GA. In a steady-state GA, only one population evolves. A new organism is immediately inserted into the population using different replacement techniques. In Figure 2 we have the general structure of a steady-state GA. After generating the initial population the algorithms iteratively applies selection, mating, mutation, and replacement operators until the end criterion is satisfied or the maximum number of steps has been reached.

The *selection operator* determines which two organisms will mate. Normally those organisms with higher fitness values are selected more often for mating by using one of several techniques like roulette wheel selection, rank base selection, and so on. These approaches are not appropriate when the search space is multimodal, because it tends to bias selection towards one of the peaks. The approach we use here is named *crowding selection* . It is similar to *mating restriction* [5], where organisms from the same peak are selected to mate. The main difference is that in crowding selection one does not restrict mating to organisms associated with the same peak, but encourages similar organisms to mate. The first parent is selected sequentially from the population, thus allowing most of the organisms from the population to reproduce. The mating partner is selected as the most similar organism from a group formed by randomly selected organisms from the population. The similarity between two organisms is determined in the decoded search space using the Euclidean distance metric. This approach, combined with the replacement method described

below, allows organisms from different peaks to coexist while maintaining diversity in the population.

The *mating operator* simulates nature's DNA replication process. The goal of mating is to create new organisms (solutions) by combining features from organisms in the population. The simplest form of mating is crossover in which a division point is chosen at random and the parents' tails (bits to the right of the division point) are interchanged as shown below.

```
Parent 1: 1 0 0 1 0 1 1 1|1 0 0 0 0 1
Parent 2: 1 0 1 0 0 1 1 0|0 1 1 1 0 1
                          |-> crossover point chosen arbitrarily
Offspring 1: 1 0 0 1 0 1 1 1|0 1 1 1 0 1
Offspring 2: 1 0 1 0 0 1 1 0|1 0 0 0 0 1
```

After the offspring are generated the *mutation operator* is applied. It consists of iterating through the bits in each offspring and changing them from 1 to 0 or 0 to 1 with a very low probability. Mutation prevents the loss of characteristics from the population, and insures that the system evolves beyond local maxima.

The crux of the replacement method is to insert the offspring into the population using a *replacement operator*. It is this operator that ensures the survival of the fittest within the *species* (organisms from the same peak) by replacing the worst organism out of a group of most similar candidates. For each child, the operator forms *crowding factor groups* containing *crowding size organisms* selected at random from the population. The organism with the lowest fitness out of the most similar representative in each group is chosen to die. An example of this operator is shown in Figure 3. The main idea behind this operator is to increase competition among the members of the same species, allowing each species to evolve to it's best. The higher the crowding size the more probable the replaced organism is from the same species. The higher the crowding factor the better the chances are that the replaced organism has lower fitness. A similar technique [6] called *enhance crowding* has been used before, where the most similar organism out of a group of worst candidates is replaced. This technique is biased toward higher peaked organisms and tends to eliminate lower peaked organisms from the population rapidly, where as our technique keeps a majority of peaks for a longer period.

This GA model was found to be the best suited for the test functions described before. In the work done by others[4,5] the operators were either too restrictive or did not differentiate between elements in different peaks. For an operator to work correctly in a dynamic multimodal space, it must balance convergence in each peak with the search for other peaks. Crowding selection and the replacement method described above seem to achieve this balance.

Population	Fitness	Random Groups	Euclidean Distance	Similar Individuals
1011110101	15	1011110101	17	
0100010110	22	1011100111	13	0010010111
0100101110	18	0010010111	10	
1011100111	13	0100101110	8	
1101011110	6	0100010001	2	0100010001
0011010100	10	1111100110	20	
0100010001	5			
1111101010	9	1011100111	13	
0010010111	19	1101011110	17	1011100111
1111100110	14	1111101010	18	

Child to be inserted is 1011011110, replaces organism 0100010001.

Figure 3: Example of replacement operator with crowding size 3 and crowding factor 3.

4. Simulation Results

4.1 Selection of Metric

Before examining the results it is necessary to describe the metrics used to measure the performance in our GA model. The online and offline performance metrics as described in [4], were used to evaluate the convergence rate of our GA model for the static functions 1, 2, and 3. These values measure the population average and the best fitness average respectively over all generations. These online and offline metrics are not appropriate for dynamic functions because the best fitness and the average fitness of the function are not maintained. To evaluate the behavior of dynamic and static multimodal functions we introduce *organisms per peak* as a new metric to measure the number of organisms per peak after each generation. All organisms within certain distance from the location of the peak were associated with that peak. Organisms in overlapping regions were counted in both peaks and those not belonging in any peak are associated with another group. This metric allowed us to examine the competition between species from different peaks.

4.2 Genetic Algorithm Parameters

The GA parameters were set as follows:

Population size: 200 or more according to the number of peaks
Crossover probability: 0.95
Mutation probability: 0.06
Crowding selection group size: 4
Crowding Factor: 5
Crowding group size: 10
Number of chromosomes: 2
Bits per chromosomes: 30

These parameters were selected after various attempts and shown to work very well with the functions given above. For a particular function there might be a better combination of these parameters. Most of the time the GA was allowed to run for about 150 generations and then the results examined. In some cases, more generations were allowed in order to observe the effect of generations on the diversity of the population.

4.3 Results

The GA model performed very well for F1; each peak maintained similar number of organisms during all the generations. The peaks (see Figure 1c) have a height of 100, width of 0.0004, and located at (45000, 2000) and (15000, 62000). The coordinates of the search space were defined to be in the range (0, 65535) in both x and y directions. We observed an average of 66 organisms in peak 1 and 70 organisms in peak 2 with a variance of $\pm$ 11 from generation 5 to generation 150. Figure 4 contains the online and offline performance for F1. The offline and online performance for all the static functions were very similar. The performance in both cases increased almost monotonically in each generation. This occurs because the weak organisms in each peak are mostly replaced by better organisms and the best performer in each peak always survived. Analogous figures for F2 and F3 are not shown. Note that there is a period where the performance is low and then it increases rapidly until it stabilizes. The jump occurs when the GA finds very good representatives from the peaks. We also noted that even though the organisms are converging to the peaks, there were an average of 64 organisms not belonging to any of the peaks thus preserving some diversity in the population. This occurs because organisms from different peaks are allowed to mate, thus creating some organisms not belonging to any of the peaks.

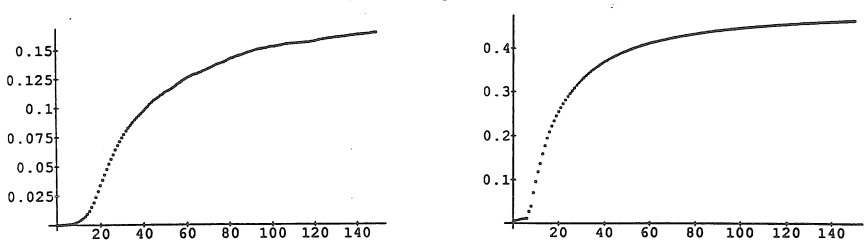


Figure 4: F1 a) online performance and b) offline performance

For the test functions with multiple height peaks, like F2 (Figure 1b) and F3 (Figure 1c), our GA model maintains a balanced distribution of the organisms between all peaks during the first 5-10 generations. After that those peaks with larger heights will start to dominate over the other peaks, but will not takeover the complete population. After many generations the number of organisms per peak will settle with shorter peaks having fewer organisms and higher peaks with more organisms. F3 has five peaks with the following configuration:

Peak no.	Location	Height	Width
1.	(140.586, 44.5314)	0.644506	0.0775816
2.	(102.585, 16.3897)	0.536561	0.76709
3.	(141.476, 5.18466)	0.39551	0.741381
4.	(153.506, 40.8626)	0.514533	0.799701
5.	(139.351, 43.8705)	0.889262	0.21302

Note that peaks 1 and 5 are very close that they practically merge to form the highest peak. Table 5 shows the organisms per peak for F3. Note also that the tallest peak, namely peak 5 has the maximum number of organisms after the initial has subside. Figure 6 shows snapshots of the location of the organisms for F2 at different generations. The location of most of the peaks can be seen from the graphs. Even after 100 generations a few organisms exist in the lower peaks.

Table 5: Organisms per peak for F3

Generation	Peak 1	Peak 2	Peak 3	Peak 4	Peak 5	Others
0	11	8	15	19	11	148
10	72	48	34	22	72	29
20	62	52	46	16	62	26
30	71	53	43	7	71	29
40	66	54	40	10	66	34
50	67	52	48	9	66	28
60	64	61	38	9	64	31
70	69	48	45	4	69	36
80	59	52	46	9	60	33
90	67	57	41	9	67	31
100	67	60	43	5	67	26
110	64	58	38	7	65	38
120	70	58	46	8	70	22
130	71	52	37	12	69	32
140	64	52	49	6	64	32
150	60	64	46	6	57	26

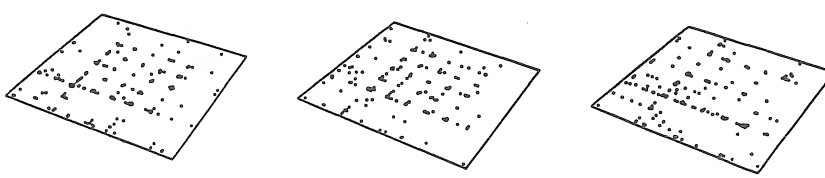


Figure 6: Location of organisms in F2 at generations 25, 50, and 100.

Results for F4 were encouraging as well. F4 was defined as a * F3 + b * G, where a varies from 0.0 to 1.0 and b from 1.0 to 0.0. F3 was defined above and G is a function similar to F3 with the following parameters:

Peak no.	Location	Height	Width
1.	(129.371, 10.0084)	0.907779	0.634976

2.	(109.307, 41.8726)	0.0708946	0.789199
3.	(127.675, 19.0206)	0.133323	0.886113
4.	(159.906, 58.8545)	0.806377	0.472369
5.	(139.827, 20.2571)	0.912816	0.166007

We ran our GA model with *a* and *b* changing 1% every generation and 1% every 10 generations. In both cases the GA was able to find the peaks of F3 during the first generations and then the peaks of G as the value of *b* kept increasing. Table 7 shows the organisms per peak for F4 changing 1% every generation. Note that the GA was able to maintain organisms in every peak most of the time.

Table 7: Organisms per peak for F4.

	Function F3					Function G					
Gen	Peak 1	Peak 2	Peak 3	Peak 4	Peak 5	Peak 1	Peak 2	Peak 3	Peak 4	Peak 5	Other
0	11	8	15	19	11	14	16	15	13	17	87
10	76	49	41	12	76	3	4	3	2	2	16
20	66	55	37	11	65	7	7	6	0	9	21
30	65	50	46	5	65	5	6	4	2	9	18
40	57	46	46	5	57	9	2	6	11	7	24
50	52	49	44	4	52	5	3	4	29	6	13
60	51	46	7	5	51	2	3	7	38	33	15
70	22	41	4	2	22	1	5	5	59	48	17
80	9	36	5	2	9	4	1	7	65	66	14
90	18	33	7	4	18	7	1	10	56	67	12
100	7	14	3	2	7	16	0	22	67	69	19

5. Comments and Conclusions

The GA model described above was shown to find solutions in both static and dynamic multimodal functions. For each case the GA model was able to identify the region where peaks are located and create a balanced environment between organisms from different peaks while at the same time converge to the top of peaks. The number of organisms in each peak was determined by the shape of the peaks. Competition between organisms in different peaks occurred when there was little room for improvement at the top of the peaks. Higher peaks will dominate in the population.

The results show our GA model as a viable tool for finding multiple solutions in a multimodal search space. We plan to applied our model to a combinatorial problem with multiple solutions to examine its behavior in such a complex space.

References

[1] L. Bodin, B. Golden, A. Assad, and M. Ball, "Routing and Scheduling of Vehicles and Crews, The State of The Art", *Computers and Operations Research* 10 (2), 63-211 1983.

[2] M. F. Bramlette, "Initialization, Mutation and Selection Methods in Genetic Algorithms for Function Optimization", *Proceedings of the Fourth International Conference on Genetic Algorithms*, R. K. Belew and L. B. Booker, eds., 100-107, Morgan Kaufmann Publishers San Mateo, CA, June 1991.

[3] L. Davis, (ed.), *Handbook of Genetic Algorithms*, Van Nostrand Reinhold New York, NY, 1991.

[4] K. A. De Jong, *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*, Doctoral dissertation, University of Michigan, Dissertation Abstracts International 36(10), 5140B, 1975.

[5] K. Deb and D. E. Goldberg, "An Investigation of Niche and Species Formation in Genetic Function Optimization", *Proceedings of the Third International Conference on Genetic Algorithms*, J. D. Schaffer, ed., 42-50, Morgan Kaufmann Publishers San Mateo, CA, June 1989.

[6] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Reading MA, 1989.

[7] D. E. Goldberg and J. Richardson, "Genetic Algorithms with Sharing for Multimodal Function Optimization", *Proceedings of the Second International Conference on Genetic Algorithms*, J. J. Grefenstette, ed., 41-49, Lawrence Erlbaum Associates, Hillsdale, NJ, June 1987.

[8] J. J. Grefenstette, "Optimization of Control Parameters for Genetic Algorithms", *IEEE Transactions on Systems, Man and Cybernetics*, SMC-16, no. 1, January-February 1986.

[9] J. H. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, 1975.

[10] J. D. Schaffer, "A Study of Control Parameters Affecting Online Performance of Genetic Algorithms for Function Optimization", *Proceedings of the Third International Conference on Genetic Algorithms*, J. D. Schaffer, ed., 51-60, Morgan Kaufmann Publishers San Mateo, CA, June 1989.