IMPROVING GA PERFORMANCE BY USING MAXIMAL HYPER-RECTANGLE ANALYSIS AND RELATIVE FITNESS

by

CHONGSHAN ZHANG

(Under the Direction of Khaled Rasheed)

ABSTRACT

In this thesis, we present two techniques to improve the performance of the genetic algorithm (GA). First we use Maximal Hyper-Rectangle (MHR) analysis to improve GA search reliability. We propose a method to find a sufficiently large MHR for new individual insertion in GA with polynomial computational complexity. Second, we propose an idea of relative fitness to improve increasing the convergence and searching more space. The individuals are selected for reproduction according to both of their global rank and regional rank. We apply the two techniques to some GA problems, and the results demonstrate their merits.

INDEX WORDS: Genetic Algorithms, Optimization, Maximal Hyper-Rectangle, Regional Rank, Relative Fitness.

IMPROVING GA PERFORMANCE BY USING MAXIMAL HYPER-RECTANGLE ANALYSIS AND RELATIVE FITNESS

by

CHONGSHAN ZHANG

Bachelor of Science, Beijing University, People's Republic of China, 1988

Master of Science, Beijing University, People's Republic of China, 1991

A Thesis Submitted to the Graduate Faculty of The University of Georgia in Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE

ATHENS, GEORGIA

2005

IMPROVING GA PERFORMANCE BY USING MAXIMAL HYPER-RECTANGLE ANALYSIS AND RELATIVE FITNESS

by

CHONGSHAN ZHANG

Major Professor:	Khaled Rasheed
Committee:	Liming Cai Thiab Taha

Electronic Version Approved:

Maureen Grasso
Dean of the Graduate School
The University of Georgia
May 2005

DEDICATION

This is dedicated to my loving wife Ruoyan, Zhang.

ACKNOWLEDGEMENTS

I am very thankful for the members of my thesis committee. I want to express my thanks to Dr. Taha and Dr. Cai for their time reviewing my thesis and their insightful advices. Many thanks to my advisor, Dr. Rasheed, to whom I owe the most overwhelming debt of gratitude for his support and his invaluable guidance for my thesis research.

TABLE OF CONTENTS

Page

LIST OF FIGURES

Page

CHAPTER 1

INTRODUCTION AND LITERATURE REVIEW

The genetic algorithm (GA) is based on Darwin's evolution theory. In nature, individuals that better fit the environment are more likely to survive, breed and pass their characteristics on to the future generations. In a typical GA, first an initial population is created containing a number of individuals. Each individual has an associated fitness measure, typically representing an objective value. Only individuals with relatively high fitness are selected to produce offspring by crossover, mutation and/or other operators, while those with lower fitness will get discarded from the population. The result is another set of individuals based on the previous population, usually leading to subsequent populations with better average fitness.

The GA approach has repeatedly proven to be a robust search and optimization method in numerous theoretical and practical domains. However, there are some potential problems with the GA.

One of them is that it is not easy to evaluate the confidence level as to whether a GA run may have missed a complete area of good points, and whether the global optimum was found. We accept this but hope to improve the confidence in our results by showing that no large gaps are left unvisited in the search space. One way to reduce the risk is to ensure that some individuals are created randomly in every generation. It is usually observed that as the search progresses, the population gets gradually concentrated

in one or a few regions which may or may not contain the global optimum. The capability of exploring new space will be weaker. Niching methods [1] attempt to search more space using fitness sharing, crowding or other diversity promotion techniques. The problem with these methods is that they are implicit rather than explicit in their attempt to cover the search space. Despite the use of niching methods, there could be large regions of the search space that are never explored.

To overcome these drawbacks, our idea is to create some individuals directly in each generation, placing them in big empty spaces to promote exploration. In doing this, the convergence may be slightly slower but the probability of premature convergence to local sub-optima is significantly reduced. By inserting some individuals in big empty spaces, our confidence in the results increases because no large gaps exist in the search space. However it is not easy to find the biggest empty space, particularly in multi-dimensional problems. Fortunately, however, for a GA problem, it is not necessary to find the exact biggest empty space; a sufficiently large empty space is good enough to insert new individuals. In Chapter 2, we present a method to find a sufficiently large empty space for new individual insertion in a GA with polynomial computational complexity. We also apply it to a number of real GA problems to check its performance.

Another problem is how to trade off improving the convergence and searching more space. The traditional GA with elitist selection is suitable for locating the optimum of unimodal functions as the population converges to a single solution in the search space. Real GA problems, however, often lead to multimodal domains and so require the identification of multiple optima, either global or local.

There are numerous approaches for attempting to find a diverse set of good solutions for a multimodal problem. Two of the most successful approaches are fitness sharing and crowding. Fitness sharing was originally introduced by Holland [2] and improved by Goldberg and Richardson [3]. The idea behind a fitness sharing scheme is that, if similar individuals are required to share fitness, then the number of individuals that can reside in any one portion of the fitness landscape is limited by the fitness of that portion of the landscape. Sharing results in individuals being allocated to optimal regions of the fitness landscape. This method can locate and maintain multiple solutions from different peaks within a population, however, a GA under sharing will not sufficiently converge population elements atop the peaks it locates. The crowding method [4, 5] is based on the idea of preserving diversity by ensuring that new individuals replaced similar members of the population. The number of individuals congregating about a peak is largely determined by the size of that peak's basin of attraction. The crowding method also can locate and maintain multiple solutions within a population. However, some good solutions may be ignored if the sizes of their basins of attraction are small.

We wish to find a good scheme that can find multiple good solutions, converge to a peak in each basin of attraction, and make the concentration near the peaks with higher fitness higher. In Chapter 3, we propose the idea of "relative fitness" to approach this goal. When individuals are selected for reproduction, we not only consider their fitness, but also consider their neighbors' fitness. We apply this idea to a number of GA examples and compare its performance with traditional GA and some niching methods.

CHAPTER 2

IMPROVING GA SEARCH RELIABILITY USING MAXIMAL HYPER-RECTANGLE ANALYSIS[1]

[1] Chongshan Zhang, Khaled Rasheed, Accepted by The Genetic and Evolutionary Computation Conference (GECCO-2005), nominated as best papers.
Reprinted here with permission of publisher, 04/26/2005.

2.0 ABSTRACT

In Genetic algorithms it is not easy to evaluate the confidence level in whether a GA run may have missed a complete area of good points, and whether the global optimum was found. We accept this but hope to measure some degree of confidence in our results by showing that no large gaps were left unvisited in the search space. This can be achieved to some extent by inserting new individuals in big empty spaces. However it is not easy to find the biggest empty spaces, particularly in multi-dimensional problems. For a GA problem, however, it is not necessary to find the exact biggest empty spaces, a sufficiently large empty space is good enough to insert new individuals. In this paper, we present a method to find a sufficiently large empty space for new individual insertion in a GA while keeping the computational complexity as a polynomial function. Its merit was demonstrated in several domains.

Categories and Subject Descriptors

I.2.8 [**Computing Methodologies**]: Artificial Intelligence – *problem solving, control method, and search.*

General Terms

Algorithm, Performance.

Keywords

Genetic Algorithms, Optimization, Maximal Hyper-Rectangle.

2.1 INTRODUCTION

Genetic algorithm [9] search is a probabilistic search approach which is founded on the ideas of evolutionary processes. The GA procedure is based on the Darwinian principle of survival of the fittest. For the general GA, first an initial population is created containing a number of individuals. Each individual has an associated fitness measure, typically representing an objective value. Only individuals with relatively high fitness are selected to reproduce offspring by crossover and mutation, while those with lower fitness will get discarded from the population. The result is another set of individuals based on the previous population, leading to subsequent populations with better individual fitness in most cases. The GA approach has repeatedly proven to be a robust search and optimization method in numerous theoretical and practical domains.

However there are potential problems. If we strictly adhere to the Darwinian evolution paradigm, the population in every generation only weakly depends on the initial population, because all offspring are created from their parents. However, only the first generation is created randomly. It is inevitable as the GA converges that exploitation will replace the initial exploration. Nevertheless, it is hard to guarantee that this will not happen too soon leading to premature convergence. One way to reduce the risk is to ensure that some individuals are created randomly in every generation. It is usually observed that as the search progresses, the population gets gradually concentrated in one or a few regions which may or may not contain the global optimum. The capability of exploring new space will be weaker. Niching methods [8] attempt to reduce the risk using sharing, crowding or other diversity promotion techniques. The problem with these methods is that they are implicit rather than explicit in their attempt to cover the search

space. Despite the use of niching methods, there could be large regions of the search space that are never explored. To overcome these drawbacks, our idea is to create some individuals directly in each generation, placing them in big empty spaces to promote exploration. In doing this, the convergence may be slightly slower but the probability of premature convergence to local sub-optima is significantly reduced. By inserting some individuals in big empty spaces, our confidence in the results increases because no large gaps exist in the search space.

However, finding the biggest empty space in a multi-dimension space is not easy. The first step is to define empty space. We can use empty hyper-sphere, empty hyper-rectangle or other shapes. For example, if we adopt empty hyper-rectangle, it is still an infinite set. Liang-ping Ku et al.[1] proposed using Maximal Hyper Rectangle (MHR) to measure the empty space and tried to find the biggest MHR. In this paper, we adopt the concept of MHR to describe the empty space (Figure 2.1), defined as follows:

(1) All the sides of MHR are parallel to the respective axis, and orthogonal to the rest.

(2) MHR does not contain any points in its interior.

(3) No other MHR exists which falls entirely within the interior of the MHR. In other words, on each surface of the MHR, there is at lease one point.

We adopt the MHR concept because it is representative of all possible empty hyper-rectangles and the number of MHRs is finite. Secondly, it is easier to handle because the space and points are usually described in the Cartesian coordinate system.

The problem of finding empty hyper-rectangles has been studied repeatedly in the literature. In [1], the computational complexity to find the biggest MHR is $O(n^{2k-1}k^3(\lg n)^2)$, where n is number of points and k is the number of dimensions in the dataset.

Liu et al.[2] motivate the use of empty space knowledge for discovering constraints. Their proposed algorithm runs in $O(n^{2(k-1)}k^3(\lg n)^2)$. Even in two dimensions this algorithm is impractical for large values of n. In an attempt to address both the time and space complexity, they proposed only maintaining maximal empty hyper-rectangles which are larger than some size. However the number of MHRs may still be very large because it is difficult to correctly set the size, a priori, in some problems. Furthermore, many MHRs are largely overlapping. Edmonds et al.[3] proposed finding all MHRs by considering each 0-entry$<x, y>$ of M one at a time, row by row, where M is an $|X|\times|Y|$ matrix (for two dimensions), X and Y denote the set of distinct values in the data set in each of the dimensions. Their proposed algorithm runs in $O(n^{2(k-1)}k)$. However, it is not applicable for continuous spaces. Other approaches have been proposed that use decision tree classifiers to approximately separate occupied space from unoccupied space, then post-process the discovered regions to determine MHRs[4]. These methods do not guarantee that all maximal empty rectangles are found.

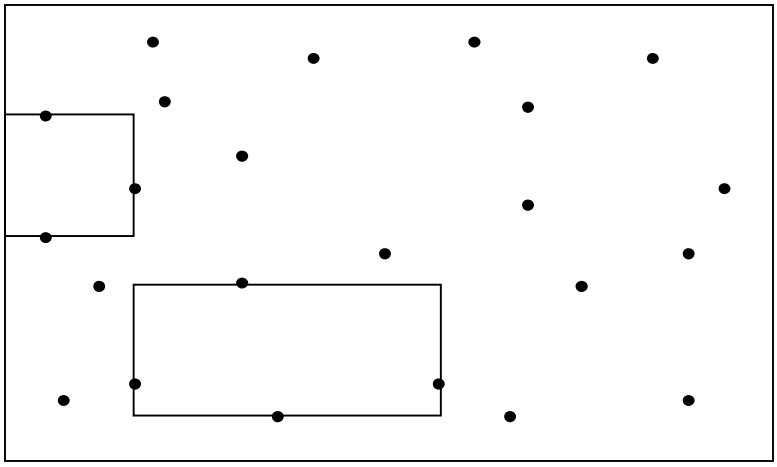

Figure 2.1 Two examples of MHRs in a 2-D space

Despite the extensive literature on this problem, none of the known algorithms are effective for large data sets. Fortunately, for the purpose of improving the GA reliability, we do not have to find the exact biggest MHR, a sufficiently big MHR is good enough.

Now the problem we will consider is as follows: Given a *k*-dimensional space S, where each dimension *r* is bounded by a minimum (S_{minr}) and a maximum (S_{maxr}) value. S contains *n* points (individuals). The problem is to find a sufficiently large MHR in which to insert a new individual.

The rest of the paper is organized as follows: In Section 2, we describe the proposed algorithm, and check the time complexity of the algorithm by setting points in S randomly and finding the calculation time for different numbers of points and dimensions. In Section 3, we apply this algorithm to some real GA problems. Section 4 presents the conclusions and future work.

2.2 THE PROPOSED APPROACH

The main idea of the algorithm is as follows: There are n points in a *k*-dimensional space S (then S has 2*k* surfaces), the middle point of two points P_i and P_j is M_{ij}. Near each M_{ij}, there is always a small empty hype-rectangle the center of which is M_{ij} and its volume is ≥ 0. We expand the empty rectangle with the same speed along all directions until some surface, for example, surface k_+, meets a point. Then along direction k_+, the expanding will stop, while along direction k_-, which is the opposite of k_+, the speed of expansion is doubled. The speed of expansion along other directions remains the same. The expansion

continues until some surface meets a point. The expansion will stop when all surface meet point (Figure 2.2).

We use a variable "time t" to measure the expansion of the empty hype-rectangle. When t=0, the volume of the empty rectangle =0, i.e. the distance between each surface and M_{ij}, d_r is 0. At the beginning, the speed along all directions, v = 1. If surface r_+ has met a point, we set v_{r+} = 0 and v_{r-} = 2. Then the distance between each surface and M_{ij}, d_r =v_r*t.

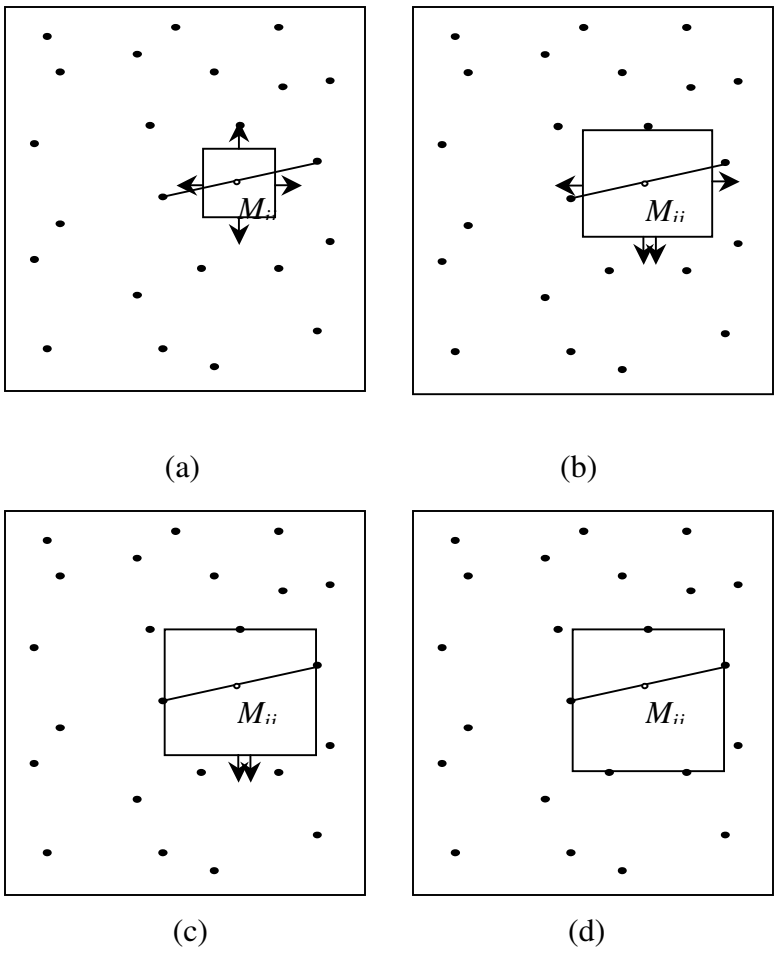

Figure 2.2 Example to illustrate the expansion process of a MHR in a 2-D space (a) an empty rectangle expands starting from a middle point; (b) one surface meets a point; (c) the other two surfaces meet two other points; (d) all surfaces meet points.

When the empty rectangle expands, we can increase t by a small amount (Δt) each time, and then check whether some surface met some point. However, if Δt is too small, we will have to check too many times. If Δt is too big, the time that some surface meets some points may lie between t and $t+\Delta t$.

To solve this problem, instead, we try to find which surface and point will meet first. With the current expansion speed, each point can be approached at some time t_i. We take P_i as an example. For each direction, we can find t_{ir} according to the distance between the current position of surface r and P_i and the speed v_{ir}. Suppose the maximum among t_{i1}, t_{i2}, …t_{i2k} is t_{ir}, then t_{ir} is the time (t_i) that the surface s of the empty rectangle expanding with the current speed meets P_i. To deal with the boundary, we add two points P_{n+1} and P_{n+2} to represent the boundary when we do the expansion, one with coordinates that are the positions of all lower surfaces of S, the other with coordinates that are the positions of all upper surfaces of S. To find the time the empty rectangle meets a surface, we consider the minimum among $t_{n+1,1}$, $t_{n+1,2}$, …$t_{n=1,k}$ rather than maximum. Suppose the minimum among t_1, t_2, …t_n, t_{n+1}, t_{n+2} is t_p, then point p is the point that the empty rectangle expanding with the current speed will first meet, and t_p is the meeting time.

After we find the point that the empty rectangle first meets, we update the position and expansion speed of all surfaces of the empty rectangle and continue the expansion process until all surfaces meet points. The expanded empty rectangle is a MHR, we call it Expanding MHR (EMHR). The EMHR_{ij} is a big MHR expanding from M_{ij}. The maximum among all EMHR_{ij} can be considered a sufficient MHR.

Although we did not investigate all MHRs because the number of MHRs is $O(n^{2k-2})$, we do expand in each MHR at least $k(2k-1)$ times because there is at least one point on each surface of each MHR.

In the course of a GA optimization, the number of points visited will increase gradually. If we run the above process every time a new point is created, the efficiency will be very low because some expansions may be repeated many times unnecessarily. In the real program, the algorithm is as follows. Given a *k*-dimensional space, we first start with no point in S. We consider all the vertices of S as initial points, so the only middle point is the center of S, and the EMHR is the whole S. We add the EMHR to the set of EMHRs. Then the n points are added to S, one point at a time. At each insertion, only the EMHRs which contain the newly added point will be updated by doing expansion from its relative middle point, because EMHRs which do not contain the newly added point will not change. We do new expansions from the middle points between the newly added point and other points, and add their EMHRs to the set of EMHRs. At each time, the above algorithm will have same results with expansions from all middle points between any two points. The MHR we require is the biggest among all EMHRs.

We do not have to keep track of all EMHRs. First, if two middle points are very close, their EMHRs will strongly overlap; thus we can discard one of them. Second, if an EMHR is very small, we can discard it. For example, if the volume of space S is *V*, the number of points so far is *N*, then if the volume of an EMHR is less than *V*/*N*, we can discard it, because it will never be the biggest EMHR.

The detailed algorithm is also described using pseudo-code (Figure 2.3).

Algorithm FindBigEmptySpace
Begin
Set the corner vertices of space S as initial points.
Set the Linked list of EMHRs (LE) = NULL.
EMHR(M) = Expending(M) // *Expansion starts from M. At this step, M is the center of S, and EMHR(M) is the whole S*
Insert(LE, EMHR(M)) // *Add EMHR(M) to LE*

When a new point P is inserted into S,
  Check LE
    If ($EMHR_i$ contains P), update $EMHR_i$
  From the mid point M_i between P and every other point *i*,
    $EMHR(M_i)$ = Expending(M_i)
    Insert(LE, $EMHR(M_i)$)
  Check LE
    If(Two EMHRs strongly overlap), delete one of them
    If($EMHR_i$ is not big enough) delete $EMHR_i$ from LE // *The above two steps may be done once every several points.*
    Return biggest EMHR and its center.
End

Expending(M)
Do (while not all dimensions of hyper-rectangle (HR) have met points)
  For each point P_i,
    $Time_i$ = getTime(P_i) // *the time on which the expanding HR meets with P_i*
  $DimAndP_j$ = minimun($Time_i$) // *the point that the expanding HR will meet first (along some dimension)*
  Update the expansion speeds.
EndDo

Figure 2.3 The pseudo-code of the proposed algorithm

We use the above algorithm to find a sufficiently large MHR to periodically add new points to a GA population. One reason is that the calculation to find the exact biggest MHR is too high, while for our purpose, a sufficiently large MHR is good enough. Another reason is that, when we try to find the big empty space in a GA trace, we may

prefer more squared space rather than a narrow space (Figure 2.4). The above expanding algorithm tends to find more squared space, because the shorter side will be meet in expanding method earlier.

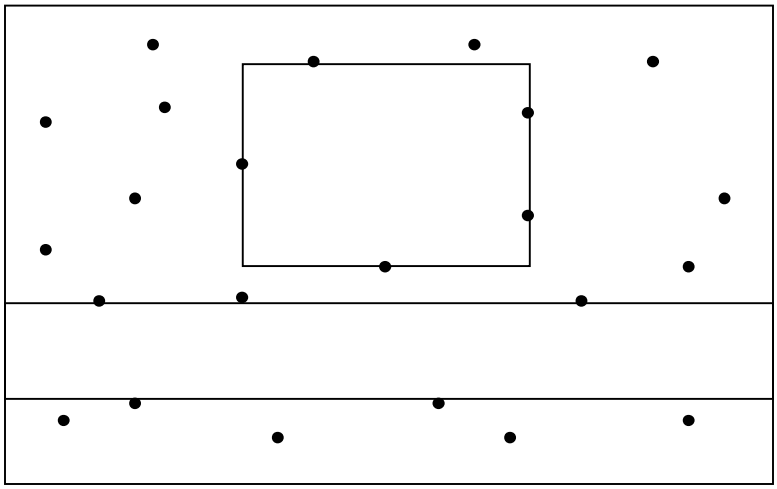

Figure 2. 4 A more squared space is more interesting empty space

Analysis of complexity: for expanding from one point, the calculation to find the next point the expanding empty rectangle will meet is done in O(nk) because in order to find the time for one point when the expanding empty rectangle meet it, we have to check all of its coordinates, i.e. O(k), and we have to find the time for all the points (n of them). The expanding will stop when all surfaces meet points, so the total calculation for expanding from one point is O(nk^2). We do expansions from all middle points between any two points, so, the total calculation is O(n^3k^2).

As pointed out earlier, we do not have to keep track of all EMHRs. If two middle points are too close, we only keep one of them and if an EMHR is too small, we discard it. The exact number of EMHRs to be discarded depends on the definitions of "too close" and "too small" but the real complexity may be lower than O(n^3k^2).

We compared the running time of the algorithm for finding the exact largest MHR and our proposed algorithm for finding a sufficiently large MHR with randomly generated data. For the sake of comparison, we find the biggest MHR by checking all MHRs, because we only try to show it is an exponential function. Figure 2.5 shows the results in terms of the actual CPU time.

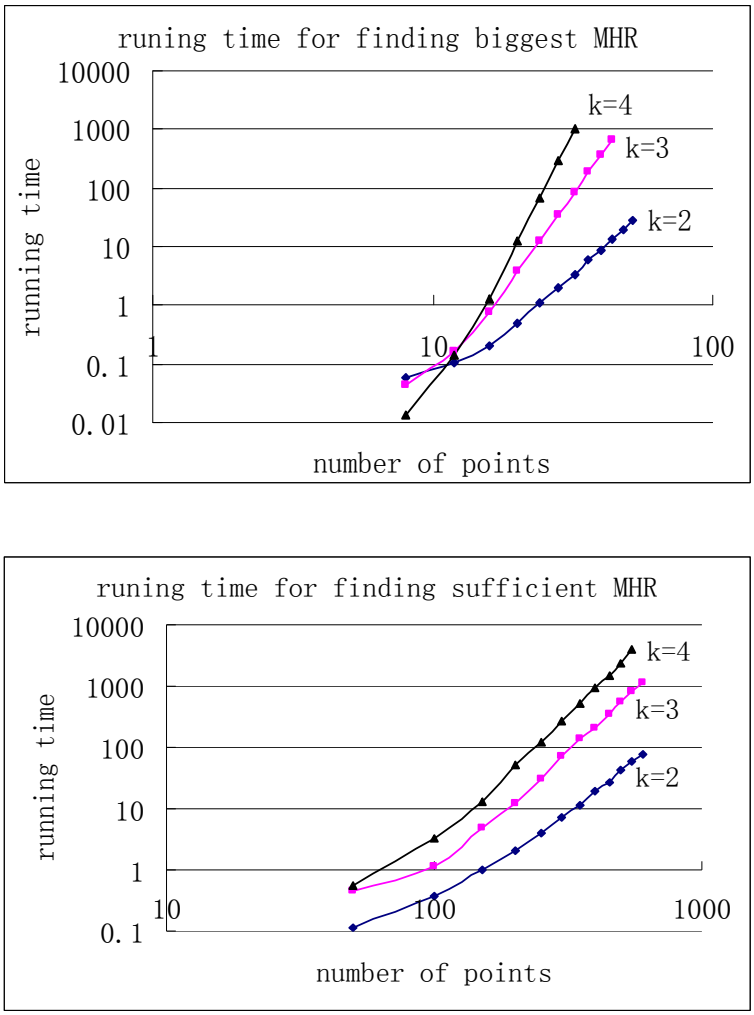

Figure 2.5 Running time for finding exact biggest MHR and sufficient big MHR

(In logarithmic scale)

For finding the exact biggest MHR for k=4 and for only 30 points, the running time is several minutes! Furthermore, even for k=4, the running time increases at a very fast rate as n increases. When n increases, for example, from 16 to 32, the running time increases more than 700 times. From Figure 2.5, we can see that the slope of the curve increases when k increases. This means that the order is a function of k. On the other hand, the running time for finding a sufficiently large MHR using our algorithm, even for several hundred points, is a few seconds. When the number of points doubled, the running time is usually increased by 8 to 10 times and not depend on k. From Figure 2.5, we can see that the slope of the curves is nearly the same for different values of k, showing that our algorithm runs in polynomial rather than exponential time.

2.3. APPLICATION TO GA OPTIMIZATION

In this section, we apply our algorithm, inserting new individuals at big empty spaces, to some GA problems, to examine its effect on performance. We consider five examples of continuous-parameter GA problems. Functions 1, 2 are taken form previous studies[5, 6, 7]. F1 is symmetric while F2 is non-symmetric and both have many local optima. The functions are shown in Figures 2.6 and 2.7. Note that we inverted the plots by adding a negative sign to the functions for the sake of clarity but all the functions used in this paper are minimized. The global optimum of F_1 is at (0, 0) and the global optimum of F2 is at (0.375, 0.375). F3 is constructed based on F1 by copying a small range including the global optimum to another location, in order to see the effect of position of global minima. Similarly F4 is constructed based on F2. F5 is constructed using F1 and F2 with each function located at different ranges, so the whole function is not continuous. The ranges

of all functions are set to $-20 \le x_1, x_2 \le 20$. In the range, each function has thousands of local optima in addition to the global optimum.

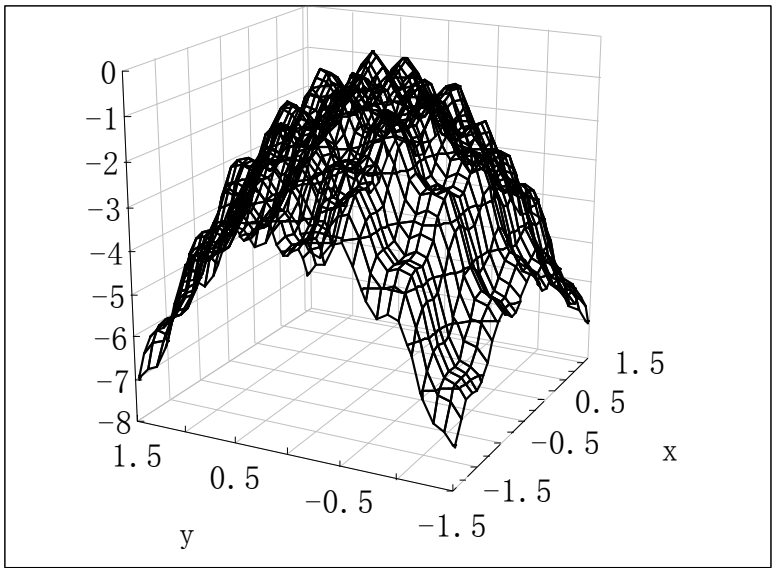

Figure 2.6 F1 $F_1(x_1, x_2) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1) - 0.4\cos(4\pi x_2) + 0.7$

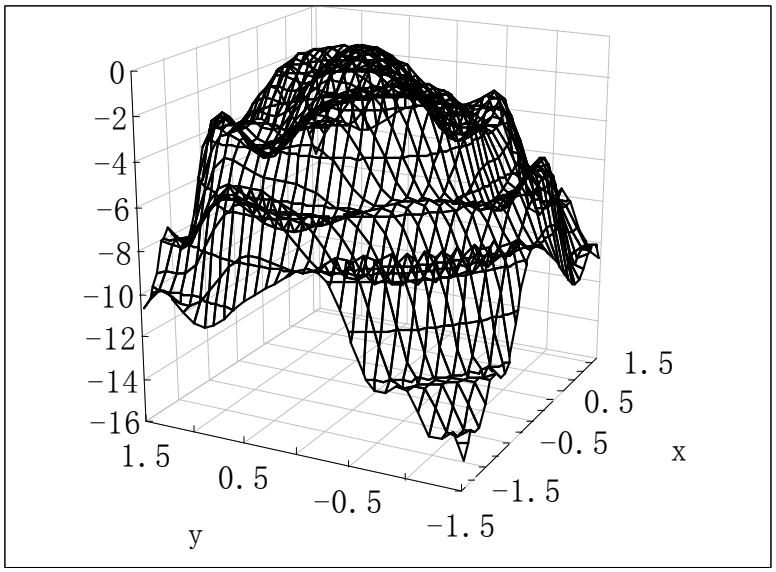

Figure 2.7 F2 With $d_1 = 0$ **and** $d_2 = 0.75$

$$F_2(x_1, x_2) = \sum_{j=1}^{2} ((x_1 - d_j)^2 + (x_1 - d_j)^2 - \cos(\pi((x_1 - d_j)^2 + (x_1 - d_j)^2)) + 1$$

The initial population is generated randomly. Rank based selection is used as it is more robust in general. Then the population enters the following loop with fixed population size *N*.

Selection: part of the population ($r_s \cdot N$, where r_s<1, is the ratio of selection) are selected to propagate to the next generation and will also be allowed to produce offspring. The selection probability of an individual is $p_i = (rank_i)^{-c} / \sum_{i=1}^{N} (rank_i)^{-c}$, where c is a constant. The individual with highest fitness is guaranteed to be selected.

Crossover: part of the population ($r_c \cdot N$, where r_c<1, is the ratio of crossover) is created by crossover of two parents selected randomly from the above selected population. The crossover operator is $x_i = x_{i,1} + R*(x_{i,2} - x_{i,1}),$ if $x_{i,2} > x_{i,1}$, $i = 1,2$, where R is a random number.

New individual insertions (by our method): the rest of the population ($(1 - r_s - r_c)N$) is created at big empty spaces, which are found according to our algorithm using all the points visited so far during the course of the optimization. The largest MHR is selected and a point is created at its center. This is repeated for the required number of individuals.

Mutation: each of the individuals in the selected population mutated with some probability P_m. This is done after the crossover step so if they took part in a crossover, it was with their original genes rather then their mutated forms. The mutation operator is non-uniform mutation[9] with $\Delta(t, y) = y \cdot R \cdot (1 - t/T)^b$, where y is the distance between the individual and the boundary, R is random number, t is the generation number, T is the

maximum generation number, b is a system parameter determining the degree of non-uniformity.

In this paper, the parameters are set as r_s =50%, r_c=30% (in the comparisons when no insertions are done this becomes 50%), P_m=0.2, c=0.5, and b=2. We use a population size of 10 and terminate the process after 50 generations.

With the insertion of new individuals at big empty spaces, we expected more space to be covered. In other words, at the end of the GA process, the empty spaces should be smaller than without insertion. Indeed the experiments supported this expectation. Figure 2.8 shows the total individual distribution for F1. F2 has similar behavior.

Figure 2.8 shows that with the insertion of new individuals, the distribution of total points visited throughout the optimization is more uniform than that without insertion. After 50 generations, without inserting individuals at big empty spaces, the biggest empty square is 7.4×7.4, while with individual insertion; the biggest empty square is 3.0x3.0. Thus more space is searched by using our algorithm.

One way to measure the success of a GA run is by checking if there are individuals within a tight ε-neighborhood of the optimum. In this paper, we consider the GA run to be successful if the individual with highest fitness is within the basin of attraction of the global optimum. The motivation behind this is to reduce the computation time. If the individual with highest fitness is close to the optimum and in its basin of attraction, usually it will go to the optimum gradually. The basin of attraction of F1 is $x_1^2/0.35 + x_2^2/0.3 < 1.0$, and for F2, it is $(x_1 + x_2 - 0.75)^2/0.4 + (x_1 - x_2)^2/0.5 < 1.0$.

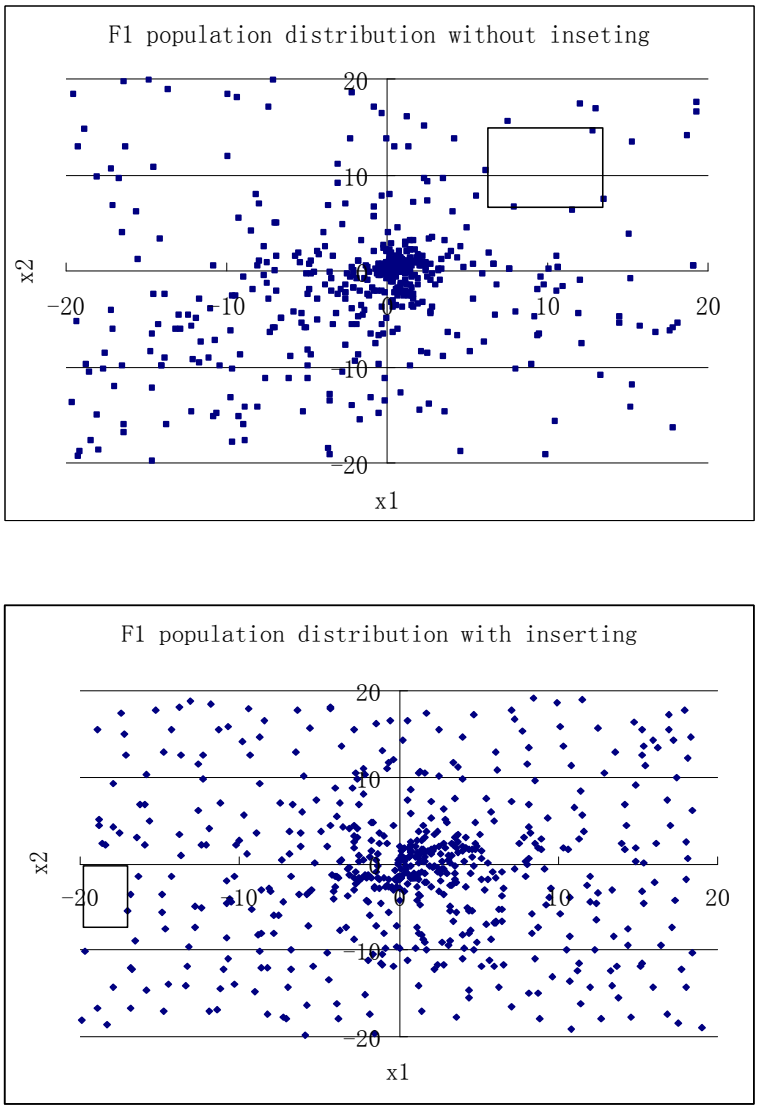

Figure 2.8 Population distribution for F1

We test the performance of our algorithm by checking how many runs are successful among 200 runs. The results are shown in Figures 2.9 and 2.10.

From Figure 2.9 and Figure 2.10, we see that our method actually degrades the performance for F1 and F2. Although this may seem surprising it is understandable.

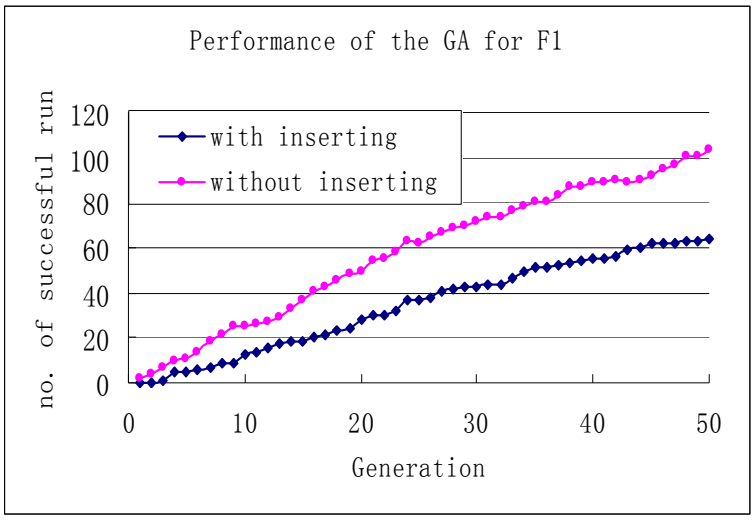

Figure 2.9 Performance of the algorithm for F1

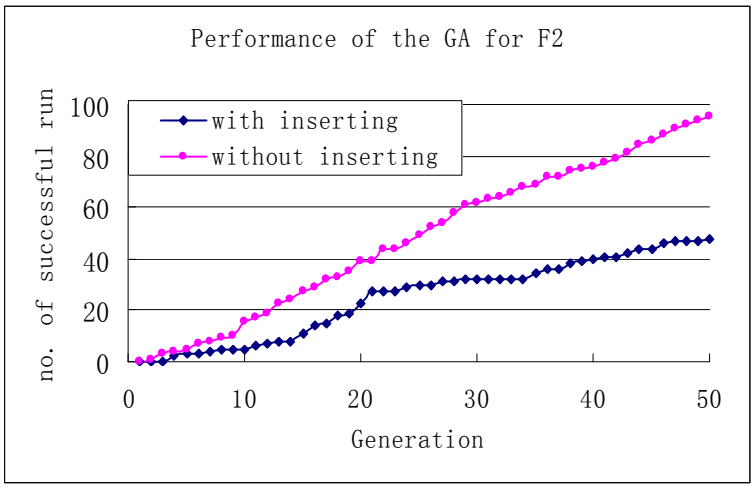

Figure 2.10 Performance of the algorithm for F2

Although F1 and F2 have many local optima, their main shapes are simple. Take F1 for example, its main shape is determined by the first and second terms of F1, $x_1^2 + x_2^2$, therefore, the values near the center (0,0) of the space is small while the values near the corner and the boundary of the space is big. The local change is caused by the third and fourth terms of F1, $-0.3\cos(3\pi x_1) - 0.4\cos(4\pi x_2)$. With the regular process of evolution,

the population will concentrate near the center gradually because of the effect of selection, crossover and mutation. The range near the boundary of the space may have big empty spaces. So if we insert some individuals in big empty spaces, with high probability they will be near the boundary rather than the center of the space. This will explore more space, however, for F1, the global optimum is at (0, 0), the insertion has no contribution to success and will degrade the performance.

However, in practice, we do not know the location of the global optimum, and the function may have any shape. So it is necessary to explore more space in order to avoid missing some space where the global optimum lies.

To demonstrate this, we constructed F3 and F4 based on F1 and F2 respectively. To create F3, we copied a small range which contains the global optimum to another location, for example, (9-11, 9-11), and add a small value to the original function to make the original global minimum a local minimum. The expression of F3 is as follows:

$$F_3(x_1, x_2) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1) - 0.4\cos(4\pi x_2) + 0.7 + 1.0$$

where $|x_i - 10| \geq 1, i = 1,2$

$$F_3(x_1, x_2) = (x_1 - 10)^2 + 2(x_2 - 10)^2 - 0.3\cos(3\pi(x_1 - 10)) - 0.4\cos(4\pi(x_2 - 10)) + 0.7$$

where $|x_i - 10| \leq 1, i = 1,2$

F4 is constructed similarly.

We then checked the performance of the algorithm for F3 and F4. The results for F3 and F4 are shown in Figures 2.11 and 2.12 respectively. The figures show that for F3

and F4 the ratio of successful runs with individual insertions in big empty spaces is higher. If the global optimum is not in the range where most values are good, inserting new individuals will increase the probability of finding it.

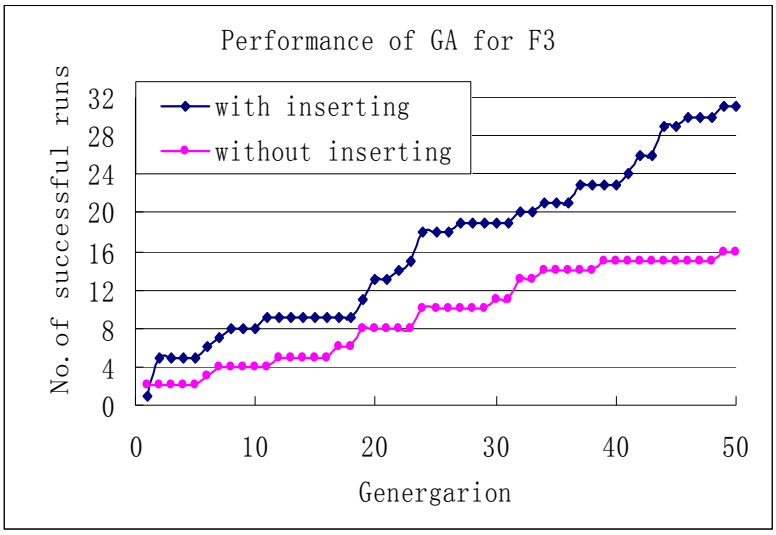

Figure 2.11 Performance of the algorithm for F3

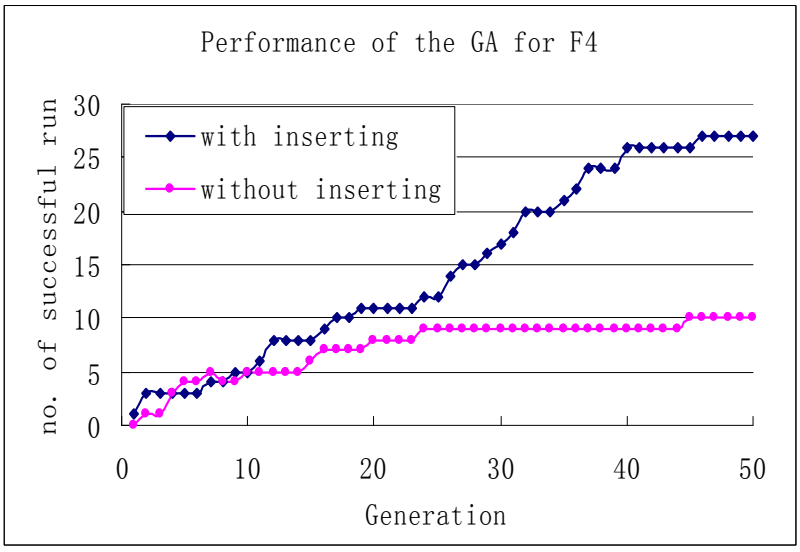

Figure 2.12 Performance of the algorithm for F4

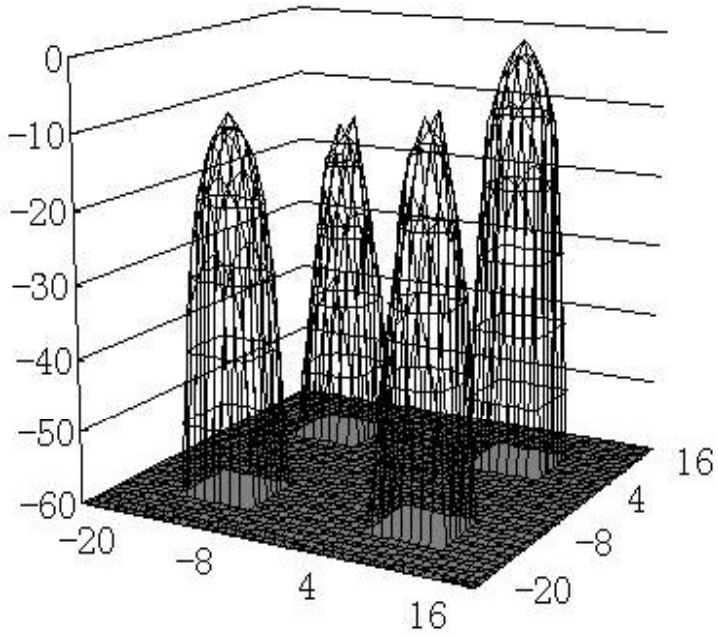

Figure 2.13 F5

We next considered a discontinuous function**.** F5 is constructed by putting F1 at the range (7-13, 7-13) with its origin at (10, 10) and the range ((-7)-(-13), (-7)-(-13)) with its origin at (-10, -10), F2 at the range ((-7)-(-13), 7-13) with its origin at (-10, 10) and the range ((7-13, (-7)-(-13)) with its origin at (10, -10). The function surface is shown in Figure 2.13. In its full range, the function is discontinuous. We further added to each continuous region a different constant (0, 4, 6, 12 respectively) so that the whole function has only one global optimum.

Figure 2.14 shows the performance of the algorithm for F5. The figure shows that for F5 the ratio of successful runs with and without insertion of new individuals in big empty space is nearly the same. It means the contribution of inserting new individual in empty space is nearly the same as the cost of inserting.

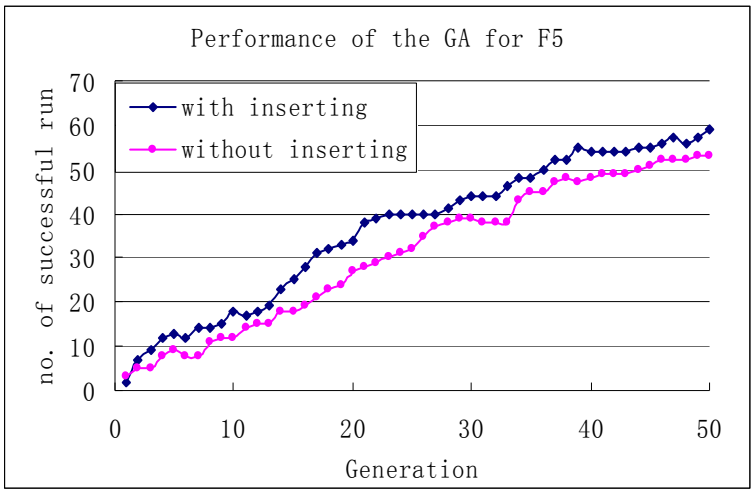

Figure 2.14 Performance of the algorithm for F5

We hypothesized that increasing the range of the peak containing the global optimum will degrade the contribution of individual insertions while decreasing that range will have the opposite effect. To test this we first increased the range with the global optimum from 6×6 to 18×18, and repeated the experiments. Figure 2.15 shows the results. We then decreased that range from 6×6 to 2×2 and the results are shown in Figure 2.16. The results clearly support our hypothesis. (Figure 2.15, 2.16).

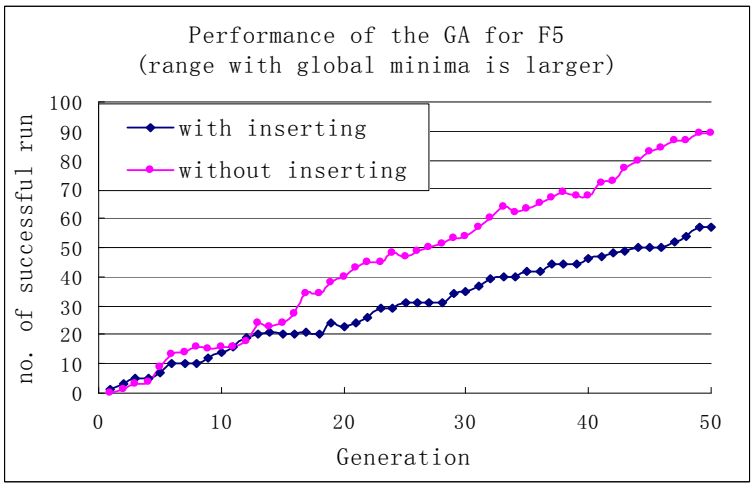

Figure 2.15 Performance of the algorithm for F5

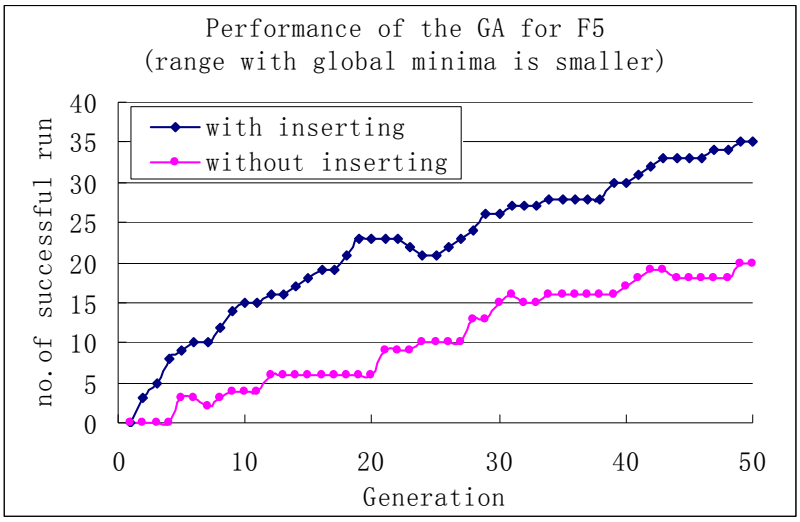

Figure 2.16 Performance of the algorithm for F5

All of the functions discussed so far were two-dimentional. Although we expected that the performance of the algorithm should be the same, we considered functions with higher dimension. F6 is constructed from F1 by expanding its dimension to four as follows,

$$F_6(x_1, x_2) = x_1^2 + 2x_2^2 - 0.3\cos(\pi x_1) - 0.4\cos(1.5\pi x_2) + x_3^2 + 2x_4^2 - 0.3\cos(\pi x_3) - 0.4\cos(1.5\pi x_4) + 1.4$$

Its basin is $x_1^2/3.1 + x_2^2/2.1 x_1^2 + x_3^2/3.1 + x_4^2/2.1 < 1.0$. F7 is constructed from F6 in the same manner we constructed F3 from F1, also for the same reason. We used a population size of 20 and terminated the process after 50 generations in this case. We tested the performance of our algorithm by checking how many runs are successful among 200 runs. The results for F6 and F7 are shown in Figures 2.17 and 2.18 respectively.

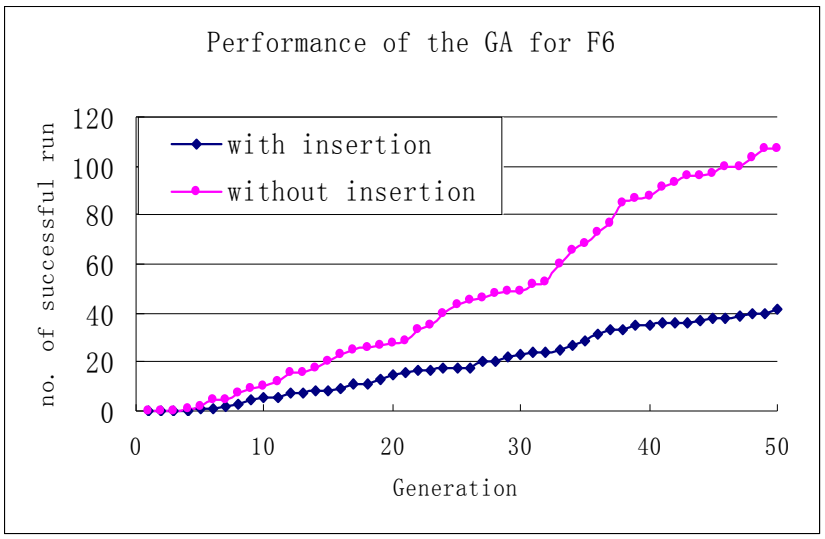

Figure 2.17 Performance of the algorithm for F6

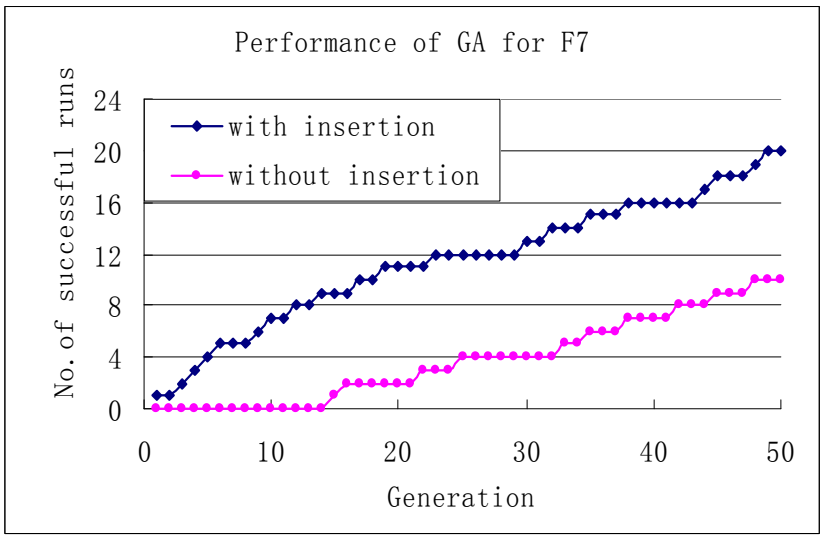

Figure 2.18 Performance of the algorithm for F7

From Figure 2.17 and 2.18, we can find that the performance for F6 and F7 is similar to F1 and F3 respectively, showing that our algorithm has same behavior for functions with higher dimension.

2.4. CONCLUSIONS

In this paper, we propose a modification for the classical Darwinian evolution metaphor commonly used in evolutionary optimization by periodically inserting new individuals at big empty spaces. To efficiently do this we propose an algorithm to find sufficiently large empty hyper-rectangles with polynomial running time. We show that it is efficient and scalable.

We conducted experiments to check its performance using several functions. The experimental results demonstrate that more spaces indeed will be searched if inserting new individuals in big empty spaces. Even when the global optimum is not located at the range where all or most points have good fitness, the GA with insertion of new individuals will have a higher probability of finding the global optimum. This is particularly useful in discontinuous and multi-modal optimization domains. The proposed method also provides a potential tool for measuring the reliability of a GA search (or any other search method for that matter) based on the size of the gaps in the search space. Further research is planned to identify such measures.

2.5. REFERENCES

[1] Ku, L., Liu, B., and Hsu, W. *Discovering Large Empty Maximal Hyper-Rectangle in Multi-Dimensional Space*. Technical Report, Department of Information Systems and Computer Science (DCOMP), National University of Singapore, 1997.

[2] Liu, B., Ku, L., and Hsu, W. *Discovering Interesting Holes in Data.* In Proceedings of IJCAI, pages930-935, Nagoya, Japan, 1997.

[3] Edmonds, J., Cryz, J., Liang, D., and Miller, R. J. *Mining for Empty Rectangles in Large Data Sets*. In Proceedings of Intl Conf on Database Theory (ICDT), pages 174--188, 2001.

[4] Liu, B., Wang, K., Mun, L.F., and Qi, X.Z. *Using Decision Tree Induction for Discovering Holes in Data.* In 5th pacific Rim International Conference on artificial Intelligence, Pages182-193, 1998.

[5] Bohchevshy, I.O., Johnson, M.E., and Stein, M.L. *Generalized Simulated Annealing for Function Optimization.* Technometrics 28(3), PP. 209-218, 1986.

[6] Fogel, D.B. *Evolutionary computation: toward a new philosophy of machine intelligence.* (Institute of Electrical and Electronics Engineers, Inc.).

[7] Ballester, P.J. and Carter J.N. *Real-parameter Genetic Algorithms for Finding Multiple optimal Solution in Multi-modal Optimization.* GECCO 2003, LNCS2723, PP. 706-717, 2003.

[8] Mahfoud, S. *A comparison of parallel and sequential Niching methods*. 6th Int. Conf. on Genetic Algorithms, pages 136-143. Morgan--Kaufmann,1996.

[9] Michalewics, , Z. *Genetic Algorithms + Data Structures = Evolution Programs*. Third Edition, Springer, 1996.

CHAPTER 3

IMPROVING GA PERFORMANCE USING RELATIVE FITNESS [2]

[2] Chongshan Zhang, Khaled Rasheed, To be submitted to the Genetic and Evolutionary Computation Conference (GECCO-2006).

3.0 ABSTRACT

The GA approach has been proven to be a robust search and optimization method in numerous theoretical and practical domains. However, there are some potential problems with GA. One of them is that when we use GA to study multimodal problems, we face a trade off between searching more regions and keeping convergence.

In this paper, we propose the idea of relative fitness to solve this problem. When individuals are selected to the next generation, we not only consider their fitness, but also consider their neighbors' fitness. One strait forward idea is to consider the global rank and regional rank together. We apply this idea to some GA examples and compare their performance with traditional GA and some niching methods. The results show that this method indeed searches more regions than traditional GA, while at the same time giving better converge in each basin of attraction.

3.1 INTRODUCTION

Genetic Algorithm (GA) [1] search is a probabilistic search approach which is founded on the ideas of evolutionary processes. The GA procedure is based on the Darwinian principle of survival of the fittest. For the traditional GA, first an initial population is created containing a number of individuals. Each individual has an associated fitness measure, typically representing an objective value. Only individuals with relatively high fitness are selected to produce offspring by crossover and mutation, while those with lower fitness will get discarded from the population. The result is another set of individuals based on the previous population, leading to subsequent populations with

better individual fitness in most cases. After some number of generations, the population should have better individual fitness values on average. The GA approach has repeatedly proven to be a robust search and optimization method in numerous theoretical and practical domains.

The traditional GA considers the entire population to act as a common gene pool, with fitness as the primary feature affecting the likelihood of an individual taking part in the creation of new offspring, and surviving to the next generation. As a result, with the process of evaluation, the individuals will usually gradually concentrate in one region of the search space.

It has repeatedly been proven that the GA is a robust method to find a good enough solution in numerous theoretical and practical domains. However, sometimes we hope to find more than one or even all good solutions for the problem.

Let's consider the evolution in the real world. Although the individuals most fit the environment (high fitness) most likely survive to next generation, the evolution of the wolves in a mountain in Asia is extremely unlikely affected by the lions in a forest in America. It means that the evolution in vivo is not only affected by the fitness, but also affected by another major parameter, namely that of the physical space, within which evolution occurs.

Ideas such as that of a global population being subdivided into smaller, infrequently communicating subpopulations, along with related concepts such as speciation and other mating restrictions, give rise to the concept of multimodal problems, i.e., problems in which there are a number of points that are better than all their neighboring solutions, but do not have as good a fitness as the globally optimal solution

(Figure 3.1). Multimodality is a typical aspect of the type of problems for which GAs are often employed, either in an attempt to locate the global optimum (particularly when a local optimum has the largest basin of attraction), or to identify a number of high-fitness solutions corresponding to various local optima.

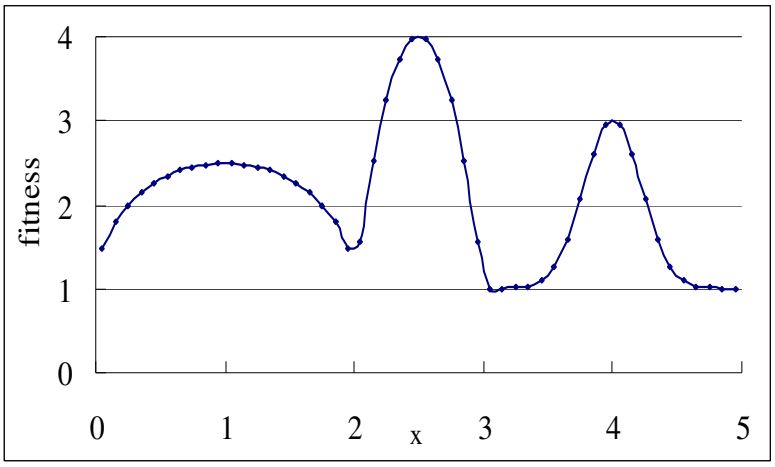

Figure 3.1 Landscape features of a multimodal problem. (There are three optima with different sizes of basins of attraction, and different "shapes".)

There are numerous approaches for attempting to find a diverse set of good solutions for a multimodal problem. Two of the most successful approaches are fitness sharing [2] and crowding [3, 4].

The idea behind a fitness sharing scheme is that, if similar individuals are required to share payoff or fitness, then the number of individuals that can reside in any one portion of the fitness landscape is limited by the fitness of that portion of the landscape. Sharing results in individuals being allocated to optimal regions of the fitness landscape. Theoretically the number of individuals residing near any peak should be proportional to the height of that peak. The fitness method can locate and maintain multiple solutions

within a population, However, A GA under sharing will not sufficiently converge population elements atop the peaks it locates.

The crowding method is based on the idea of preserving diversity by ensuring that new individuals replaced similar members of the population. The number of individuals congregating about a peak is largely determined by the size of that peak's basin of attraction. As a consequence, the individuals are easily trapped by local optima, so it may not be suitable for problems with many local optima. The crowding method also can locate and maintain multiple solutions within a population. However, some good solutions (peaks with high fitness) may be ignored if the sizes of their basins of attraction are small.

We wish to find a good scheme that can find multiple good solutions, converge to a peak in each basin of attraction, and make the concentration near the peaks with higher fitness higher. In this paper we propose a method to attempt to approach this aim.

The rest of this paper is organized as follows. Section 2 describes the proposed approaches. In section 3, we apply the proposed approaches to a multimodal example and analyze the results. In section 4, we apply the proposed approaches to an optimization problem. Finally, we present our conclusions and future work in section 5.

3.2 THE PROPOSED APPROCH

Suppose there are 10 lions in a small plain, some strong, some weak and there are 100 monkeys in the nearby wide forest, also some strong and some weak. The weakest lion is still more powerful than the strongest monkey. The interaction between individuals

among the same group is stronger than that among different groups, while also there is interaction between groups.

If we select individuals for reproduction or propagation to the next generation based only on the strength of the animal, any lion will have a higher probability to be selected than all the monkeys. Therefore we may lose the chance that a monkey may be mutated to human, which has higher fitness.

If we attempt to solve this problem by adjusting their fitness according to the population density, a common approach for fitness sharing methods, after some generations, when the density of lions is larger than that of monkeys by the same ratio that a lion is stronger than a monkey, the whole population will reach an equilibrium. Each individual will have the same shared fitness on average. The evolution in each group will nearly stop.

If we suppose that the offspring only compete with their parents, a common approach for crowding methods, the size of each group will remain the same although evolution in each group still continues. It ignores the fact that the lion is stronger than the monkey, in other words, it completely eliminates the interaction between groups.

We would like to simulate a situation in which the strongest lions have the highest probability of being selected, and the strongest monkeys have a higher selection probability than the weakest lions, because the interaction between lions and monkeys is weaker than that between lions. As a consequence the strongest lions and monkeys survive, while the density of lions is higher than monkeys.

To reach this aim, we propose that the individuals be ranked according to their global rank and regional rank (we name this double rank in order to make the description concise). The strongest lions have high global rank and regional rank; the strongest monkeys have low global rank, but have high regional rank, while the weakest lions have low global rank and regional rank. The global rank reflects the interaction between groups (also individuals), and makes the density of lions higher than that of monkeys, while the regional rank reflects the interaction between individuals in the same group, and maintains diversity. The fitness of the strongest individual in a basin of attraction does not decrease because of other individuals close to it. The evolution will never stop until it reaches the peak.

In this paper, the selection probability (new fitness) is determined by the global and regional rank as follows:

$$f(rank)_i = \frac{\lambda R_{g,i}^{-d} + (1-\lambda) R_{r,i}^{-d}}{\sum_i (\lambda R_{g,i}^{-d} + (1-\lambda) R_{r,i}^{-d})}$$

where R_g, R_r denote the global rank and the regional rank respectively, λ, d are constants, which determine the weights of the global rank and the regional rank.

We will apply the double rank method to some GA examples to investigate its properties, and compare it with traditional GAs and fitness sharing methods. We briefly introduce these methods first.

In practice the fitness sharing scheme works by considering each possible pairing of individuals i and j within the population and calculating a distance $d(i, j)$ between

them. The fitness F of each individual *i* is then adjusted according to the number of individuals falling within some pre-specified distance σ_{sh} using a power-law distribution:

$$F_{new} = \frac{F(i)}{\sum_j sh(d(i,j))}$$

where the sharing function *sh*(*d*) is a function of the distance *d* given by:

$$sh(d) = \begin{cases} 1-(dist/\sigma_{sh})^{\alpha} & \text{if } dist < \sigma_{sh} \\ 0 & \text{otherwise} \end{cases}$$

where α, σ_{sh} are constants to determine the shape of the sharing function and sharing range.

3.3 APPLICATION TO A MULTIMODAL PROBLEM

The initial population is generated randomly. Rank based selection is used as it is more robust in general. The population size remains constant. Then the population enters the following loop with fixed population size *N*.

Selection: Part of the population ($r_s \cdot N$, where r_s<1, is the selection ratio) are selected to propagate to the next generation and will also be allowed to produce offspring. The selection probability of an individual is $p_i = (rank_i)^{-c} / \sum_{i=1}^{N} (rank_i)^{-c}$, where c is a constant. The individual with highest fitness is guaranteed to be selected.

Crossover: Part of the population ($r_c \cdot N$, where r_c<1, is the ratio of crossover) is created by crossover of two parents selected randomly from the above selected

population. The crossover operator is: $x_i = x_{i,1} + R*(x_{i,2} - x_{i,1}),$ if $x_{i,2} > x_{i,1}$, $i = 1,2$, where R is a random number.

Mutation: Each of the individuals in the selected population is mutated with some probability P_m. This is done after the crossover step so if they took part in a crossover, it was with their original genes rather then their mutated forms. The mutation operator is non-uniform mutation [1] with $\Delta(t, y) = y \cdot R \cdot (1 - t/T)^b$, where y is the distance between the individual and the boundary, R is random number, t is the generation number, T is the maximum generation number, b is a system parameter determining the degree of non-uniformity. In this paper, the parameters are set as r_s =50%, r_c=50%. Pm=0.2, c=0.5, and b=2.

Let us consider the simple function shown in Figure 3.1. It consists of three parts; each part is a basin of attraction. The left part is a circle function: $fitness = 1.0 + 1.5*\sqrt{1.0 - (x - 1.0)^2}$, $0.0 < x < 2.0$, the peak is 2.5 at x = 1.0. The second part is a parabolic function: $fitness = 1.0 + 3.0*(1.0 - 4.0*(x - 2.5)^2)$, $2.0 \le x < 3.0$, the peak is 4.0 at x = 3. The third part is a normal function: $fitness = 1.0 + 2.0*\exp[-10.0*(x - 4)^2]$, $3.0 \le x < 5.0$, the peak is 3.0 at x = 4.

We apply traditional GA, fitness sharing, and the double rank method to this example, and check the distribution of the population after some generations. We use a population size of 20 and terminate the process after 50 generations. The same experiment was repeated 500 times with different random initial populations. The density of points for traditional GA, fitness sharing and double rank methods are shown in Figures 3.2, 3.3, and 3.4 respectively.

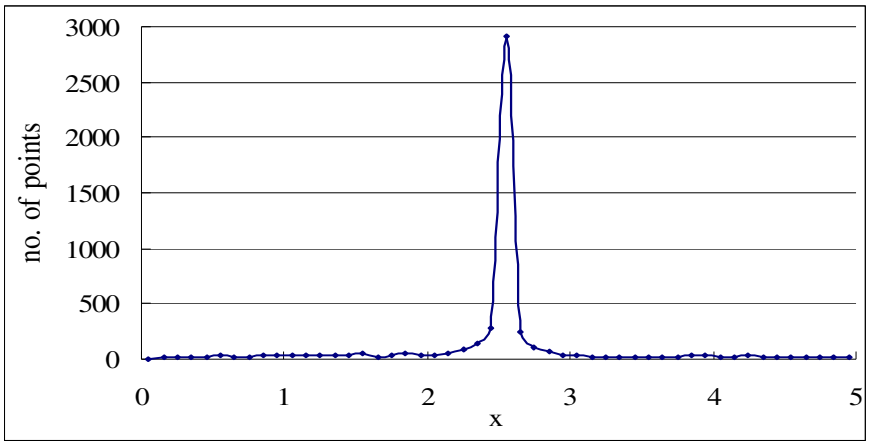

Figure 3.2 Population distribution for traditional GA

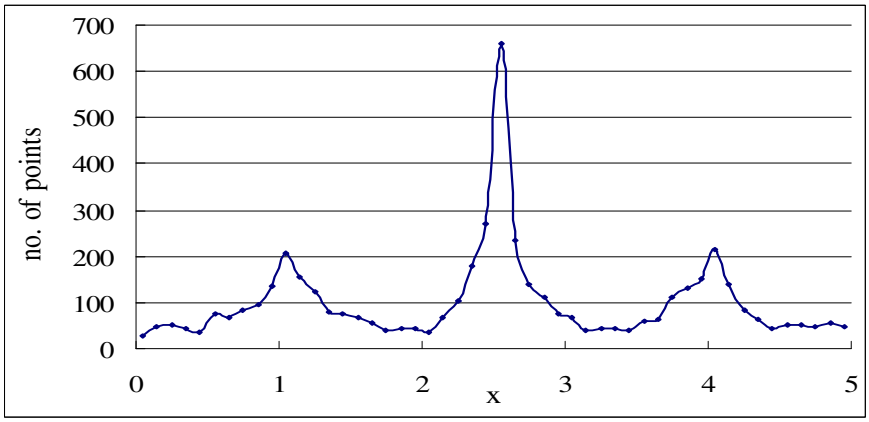

Figure 3.3 Population distribution for double rank method

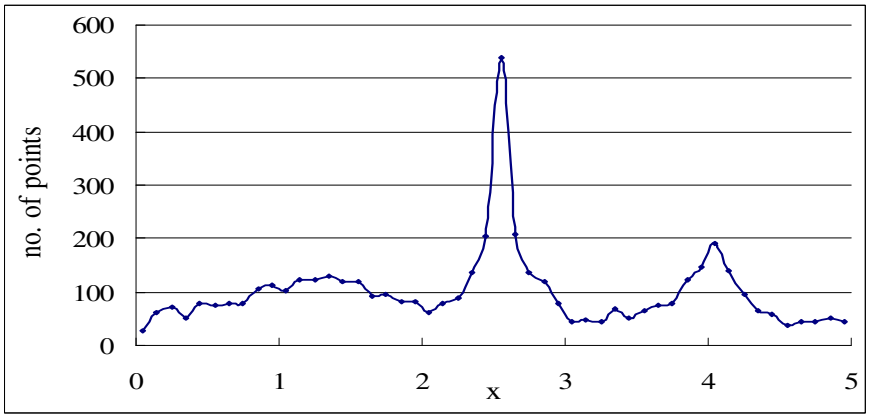

Figure 3.4 Population distribution for fitness sharing method

The above figures show the general behavior of different algorithms. For the tradition GA, the population gets gradually concentrated to the global optimum as the search progresses in this case. For the fitness sharing method, the points distribute in three basins of attraction. However in the left basins of attraction where the function is flatter than other region, the distribution of points is also flat, there is no sharp peak. It means the points in this region do not concentrate to the peak. For double rank method, comparing with traditional GA, the points distribute in all three basins of attraction rather than one region. Comparing with fitness sharing method, the density of points near the optimum in each basin of attraction is obviously high no matter the shape of the functions.

Although it is not easy to say which algorithm is better than others because an EA which excels with a given class of problems might yield poor results when applied to another class, and whether an EA performs well may depend on the interest of the researcher. There are some common judgments.

(1) For multimodal problems, whether the algorithm can locate and maintain multiple solutions.

(2) Whether the number of individuals residing near higher peaks is larger than that near lower peaks.

(3) Whether the individuals in a niche converge to the peak located in the niche quickly.

(4) Whether the results depend on the landscape features besides the fitness of peaks, such as the basins of attraction being broad or narrow, the curve of the function being flat or steep, and so on.

(5) Whether the computational complexity is high.

(6) Others, such as whether it is easy to implement, whether it is stable and so on. Usually these depend on what operators are adopted, and are not easy to generally critique.

For traditional GA, the population tends to gradually concentrate to one region with high fitness as the search progresses. Therefore it is usually used to perform optimization to find a relatively good solution, rather than to deal with multimodal problems.

Fitness sharing method can locate and maintain multiple solutions within a population, and the number of individuals residing near higher peaks is larger than that near lower peaks. However, A GA under sharing will not sufficiently converge the population elements atop the peaks it locates. The distribution of points in the left of basin of attraction in Figure 3.3 shows this clearly. Suppose after some generations, the points distribute in different basins proportionally according to the fitness of the peaks. Under this condition, sharing is not useful any more, but only hindering the points' convergence to the peaks. On the other hand, the evolution result depends on the shape of function and the size of the basin of attraction. If the function is shifted up or down, the results will be different although the problem does not really change.

For the double rank method, it can locate and maintain multiple solutions within a population like fitness sharing and crowding methods. Meanwhile, the population can

gradually concentrate near the peak in each basin of attraction, due to the effect of regional rank. Suppose the number of individuals in one region is nearly fixed after some generation, the evolution in the basin of attraction is similar to traditional GA. The number of individuals residing near higher peaks tends to be larger than that near lower peaks due to the effect of global rank. On the other hand, the results are not strongly influenced by the shape of the function, the size of basin of attraction, and whether the function shifts up or down. The mean drawback of this method is that additional calculation time is required to compute the regional rank for each individual.

For a high dimension, complex function, it is not easy to show the density of points after some generations of evolution graphically. We propose two criteria to measure its performance: (1) how close the individuals converge to the optima in each basin (2) whether the individuals converge to more regions.

Criterion (1) can be measured by the following expression:

$$conv = \sum_{i=1} (p(x_i) - f(x_i))$$

where $p(x_i)$ is the optimum value of the basin where x_i lies. The smaller *conv* is, the better the convergence of the algorithm.

We use the concept of entropy to describe criterion (2) as follows:

$$entropy = n!/\prod_i n_i! \qquad \text{or} \qquad \log(entropy) = \sum_{j=1}^{n} \log j - (\sum_i \sum_k^{n_k} \log k)$$

where n_i is the number of individuals in basin *i*. Higher entropy means the individuals distribute in more regions. The higher the entropy is, the better the performance.

F1 has 3 basins. Each basin has one optimum which can be determined. After we determine which basin the individuals belong to, we can find the difference between the

fitness and the relative optimum for each individual, then we can calculate *conv*. After we count the number of individuals in each basin, we can calculate *entropy*.

We use a population size of 40 and terminate the process after 50 generations. Same run is repeated 500 times to get the average of *conv* and *entropy*. The results are shown in Figures 3.5 and 3.6.

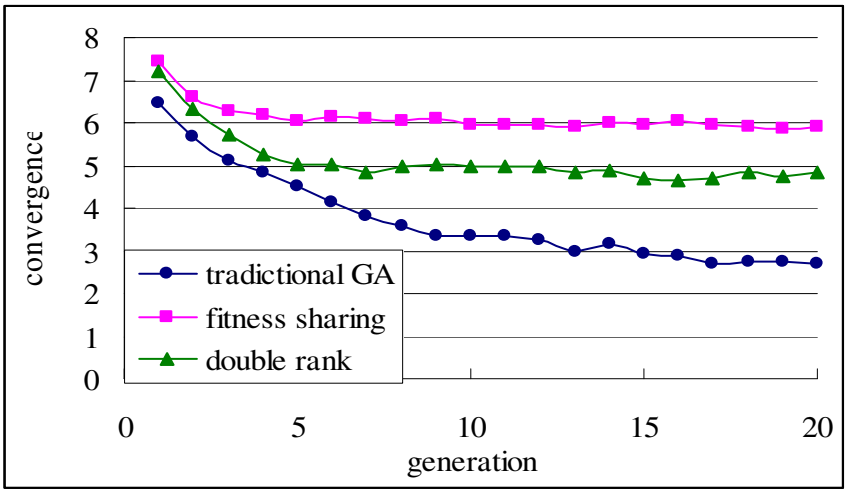

Figure 3.5 Comparison of convergence of the algorithms

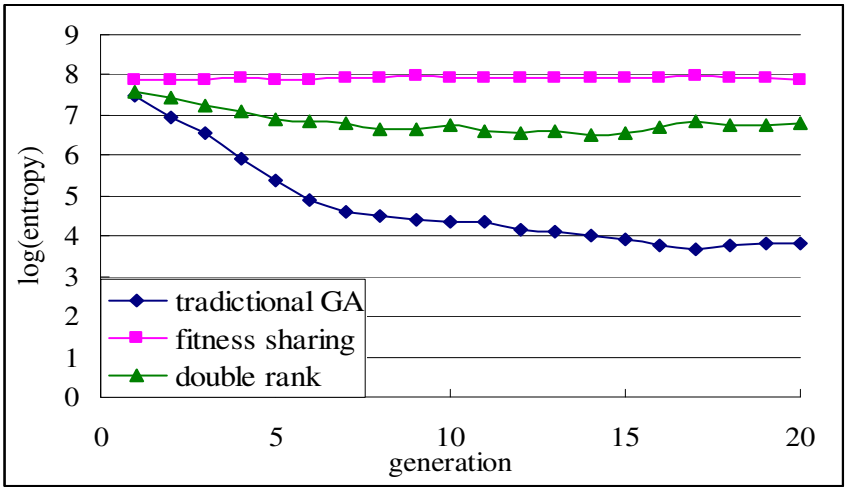

Figure 3.6 Comparison of the algorithms' ability to find more solutions

From Figures 3.5 and 3.6, we can see that with our proposed method, double rank, the individuals will locate more basins of attraction, same as fitness method. On the other hand, the individuals will get close to optima in each basin of attraction. However the convergence is slow.

3.4 APPLICATION TO AN OPTIMAIZTION PROBLEM

We have mentioned that for traditional GA, the population gets gradually concentrated to one region where the global optimum may or may not be located. In case the proposed method can locate and maintain multiple solutions within a population, we expect that if we apply the double rank method to an optimization problem, sometime we may decrease the possibility of missing the global optimum, particularly when the global optimum resides in a small basin of attraction.

To check this exception, we apply the proposed method to a continuous real-parameter GA problem (denoted as F2), which is taken from previous studies [5, 6, 7]. F2 is symmetric and has many local optima. The function is shown in Figure 3.7. Note that we inverted the plots by adding a negative sign to the functions for the sake of clarity but all the functions used in this paper are minimized. The global optimum of F1 is at (0, 0), its ranges are set to $-20 \le x_1, x_2 \le 20$. In the range, F2 has thousands of local optima in addition to the global optimum.

One way to measure the success of a GA run is by checking if there are individuals within a tight ε-neighborhood of the optimum. In this paper, we consider the GA run to be successful if the individual with highest fitness is within the basin of

attraction of the global optimum. The motivation behind this is to reduce the computation time. If the individual with highest fitness is close to the optimum and in its basin of attraction, usually it will go to the optimum gradually. The basin of attraction of F2 is $x_1^2/0.35 + x_2^2/0.3 < 1.0$.

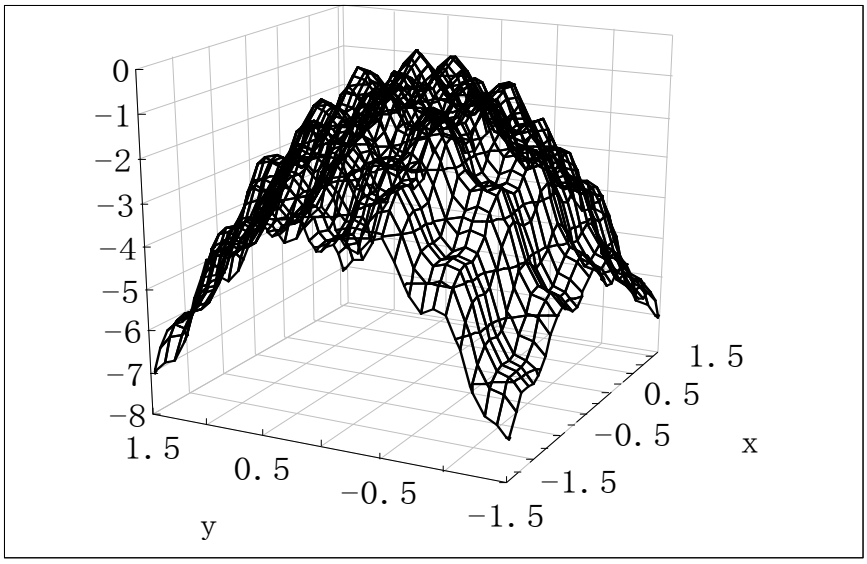

Figure 3.7 F2 $F_2(x_1, x_2) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1) - 0.4\cos(4\pi x_2) + 0.7$

We test the performance of our method by checking how many runs are successful among 200 runs. The results are shown in Figures 3.8.

From Figure 3.8, we see that our method actually degrades the performance for F2. Although this may seem surprising it is understandable. Although F2 have many local optima, their main shapes are simple. Its main shape is determined by the first and second terms, $x_1^2 + x_2^2$, therefore, the values near the center (0,0) of the space are small while the values near the corner and the boundary of the space is big. The local change is caused by the third and fourth terms, $-0.3\cos(3\pi x_1) - 0.4\cos(4\pi x_2)$. With the regular process of evolution, the population will concentrate near the center gradually because of the effect

of selection, crossover and mutation. If we select individuals using the fitness adjusted according global and regional rank, some region far away (0, 0) will be visited. This will explore more space, however, for F2, the global optimum is at (0, 0), this effect has no contribution to success and will degrade the performance. However, in practice, we do not know the location of the global optimum, and the function may have any shape. So it is necessary to explore more space in order to avoid missing some space where the global optimum lies.

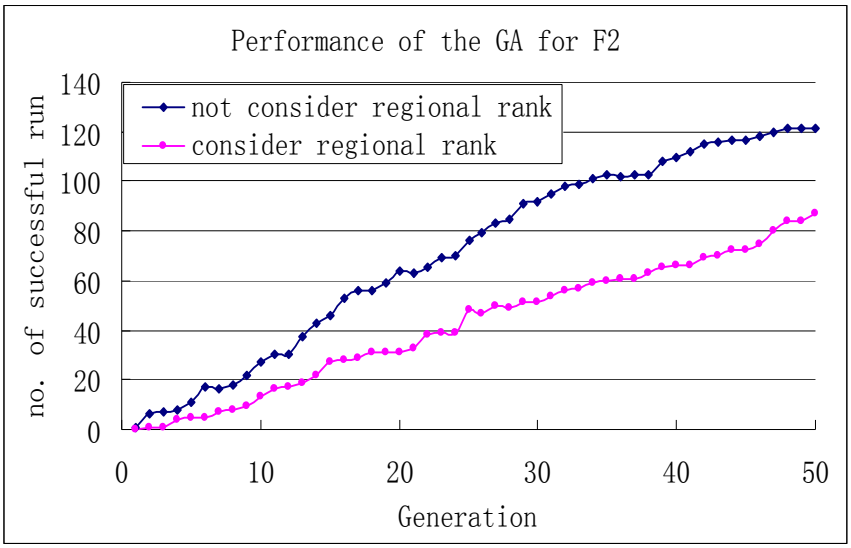

Figure 3.8 Performance of the algorithm for F2

To demonstrate this, we constructed F3 based on F2. We copied a small range which contains the global optimum to another location, for example, (9-11, 9-11), and add a small value to the original function to make the original global minimum a local minimum. The expression of F3 is as follows:

$$F_3(x_1, x_2) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1) - 0.4\cos(4\pi x_2) + 0.7 + 1.0$$

where $|x_i - 10| \geq 1$.

$$F_3(x_1, x_2) = (x_1 - 10)^2 + 2(x_2 - 10)^2 - 0.3\cos(3\pi(x_1 - 10)) - 0.4\cos(4\pi(x_2 - 10)) + 0.7$$

where $|x_i - 10| \leq 1$.

We then checked the performance of the algorithm for F3. The results for F3 are shown in Figures 3.9. Figure show that for F3 the ratio of successful runs with considering regional rank is higher. If the global optimum is not in the range where most values are good, considering global and regional rank will increase the probability of finding it.

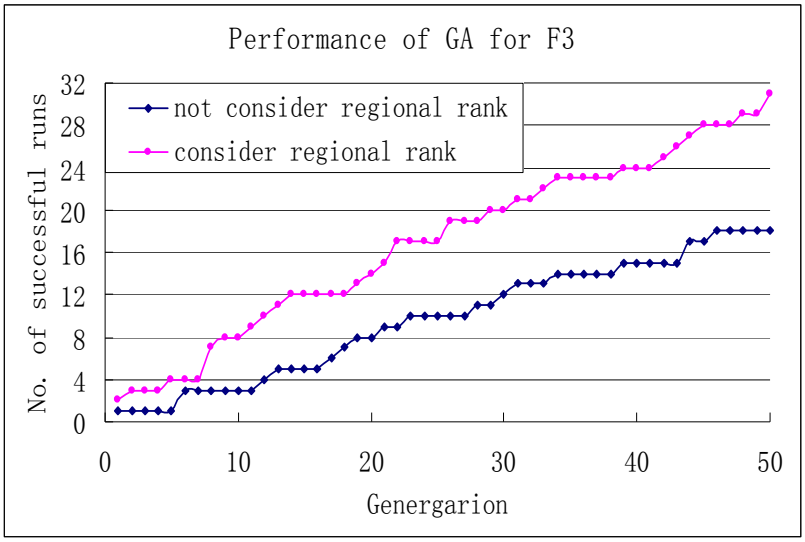

Figure 3.9 Performance of the algorithm for F3

3.5 CONCLUSIONS

In this paper, we proposed a relative fitness approach to investigate multimodal and optimization problems. When individuals are selected for reproduction or propagation to the next generation, we not only consider their global rank, but also consider their regional rank. We apply this idea to some GA examples to analyze its behavior and

compare with other algorithms. We suggest some ideas about how to measure the GA's performance. The results show that with this method the population can locate multiple solutions in multiple peaks, while still converging to optima in each basin of attraction. For some GA problems, it can also decrease the possibility of missing the global optimum.

For the double rank method, we have to choose a range in which we calculate the regional rank, however setting a reasonable range requires prior knowledge of the GA problem. For real GA problems, however, we usually have no prior information about the search space and the landscape features. On the other hand, the range relative to each individual should be different because the features of the landscape around each new individual are different. Also for double rank, inside or outside the region, the physical space of the individuals is not taken into account. To overcome these limitations, in the future, we plan to use a rank function with distance for each individual to replace double rank. The rank function with distance for an individual is the rank of the individual in the range confined by a hyper-cube the center of which is the individual and the radius of which is the distance value. After we obtained the rank function with distance for each individual, we do not have to set a range to calculate the fitness by regional and global rank, instead, we calculate the fitness by a suitable weight function. The effect of the weight function is similar to that in double rank method.

3.6 REFERENCES

[1] Michalewics, Z. *Genetic Algorithms + Data Structures = Evolution Programs*. Third Edition, Springer, 1996.

[2] Goldberg and Richardson, Genetic algorithms with sharing for multimodal function optimization. In Grefenstette, J., editor, *Proceedings of the 2nd International Conference on Genetic Algorithms and their Applications*, pages 41-49, Cambridge, MA. Lawrence Erlbaum Associates, 1987.

[3] DeJong, *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*. Ph.D. thesis, Department of Computer and Communication Sciences, University of Michigan,1975.

[4] S.W. Mahfoud, Crowding and preselection revisited. In R. Manner, B. Manderick, Eds. *Proceedings of the 2nd Conference on Parallel Problem Solving from Nature*. North-Holland, Amsterdam, 1992.

[5] Bohchevshy, I.O., Johnson, M.E., and Stein, M.L. *Generalized Simulated Annealing for Function Optimization.* Technometrics 28(3), PP. 209-218.

[6] Fogel, D.B. *Evolutionary computation: toward a new philosophy of machine intelligence.* (Institute of Electrical and Electronics Engineers, Inc.).

[7] Ballester, P.J. and Carter J.N. *Real-parameter Genetic Algorithms for Finding Multiple optimal Solution in Multi-modal Optimization.* CECCO 2003, LNCS2723, PP. 706-717, 2003.

CHAPTER 4

CONCLUSIONS

In this thesis, we present two techniques to improve the performance of the genetic algorithm (GA).

First, we propose a modification for the classical Darwinian evolution metaphor commonly used in evolutionary optimization by periodically inserting new individuals at big empty spaces. To efficiently do this we propose an algorithm to find sufficiently large empty hyper-rectangles with polynomial running time. We show that it is efficient and scalable.

We conducted experiments to check its performance using several functions. The experimental results demonstrate that more space indeed will be searched by inserting new individuals in big empty spaces. Even when the global optimum is not located at the range where all or most points have good fitness, the GA with insertion of new individuals will have a higher probability of finding the global optimum. This is particularly useful in discontinuous and multi-modal optimization domains. The proposed method also provides a potential tool for measuring the reliability of a GA search (or any other search method for that matter) based on the size of the gaps in the search space. Further research is planned to identify such measures.

Second, we proposed a relative fitness approach to investigate multimodal and optimization problems. When individuals are selected for reproduction or propagation to the next generation, we not only consider their global rank, but also consider their regional rank. We apply this idea to some GA examples to analyze its behavior and compare with other algorithms. We suggest some ideas about how to measure the GA's performance. The results show that with this method the population can locate multiple solutions, while still converging to optima in each basin of attraction. For some GA problem, it also can decrease the possibility of missing the global optimum.

REFERENCES

[1] Mahfoud, S. *A comparison of parallel and sequential Niching methods*. 6th Int. Conf. on Genetic Algorithms, pages 136-143. Morgan--Kaufmann,1996.

[2] J. H. Holland, *Adaptation in Natureal and Artificial System.* Ann Arbor, MI: Univ. Michigan Press, 1975.

[3] Goldberg and Richardson, Genetic algorithms with sharing for multimodal function optimization. In Grefenstette, J., editor, *Proceedings of the 2nd International Conference on Genetic Algorithms and their Applications*, pages 41-49, Cambridge, MA. Lawrence Erlbaum Associates, 1987.

[4] DeJong, *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*. Ph.D. thesis, Department of Computer and Communication Sciences, University of Michigan,1975.

[5] S.W. Mahfoud, Crowding and preselection revisited. In R. Manner, B. Manderick, Eds. *Proceedings of the 2nd Conference on Parallel Problem Solving from Nature*. North-Holland, Amsterdam, 1992.