**Neural Networks Problem:**

For this assignment, I changed another dataset to work with, because there is an error in MATLAB to read my original dataset. The new dataset is the Pima Indian diabetes data set from UCL machine learning dataset website. pimatest.csv is the validation/test data set, and pimatrain.csv is the training data set.

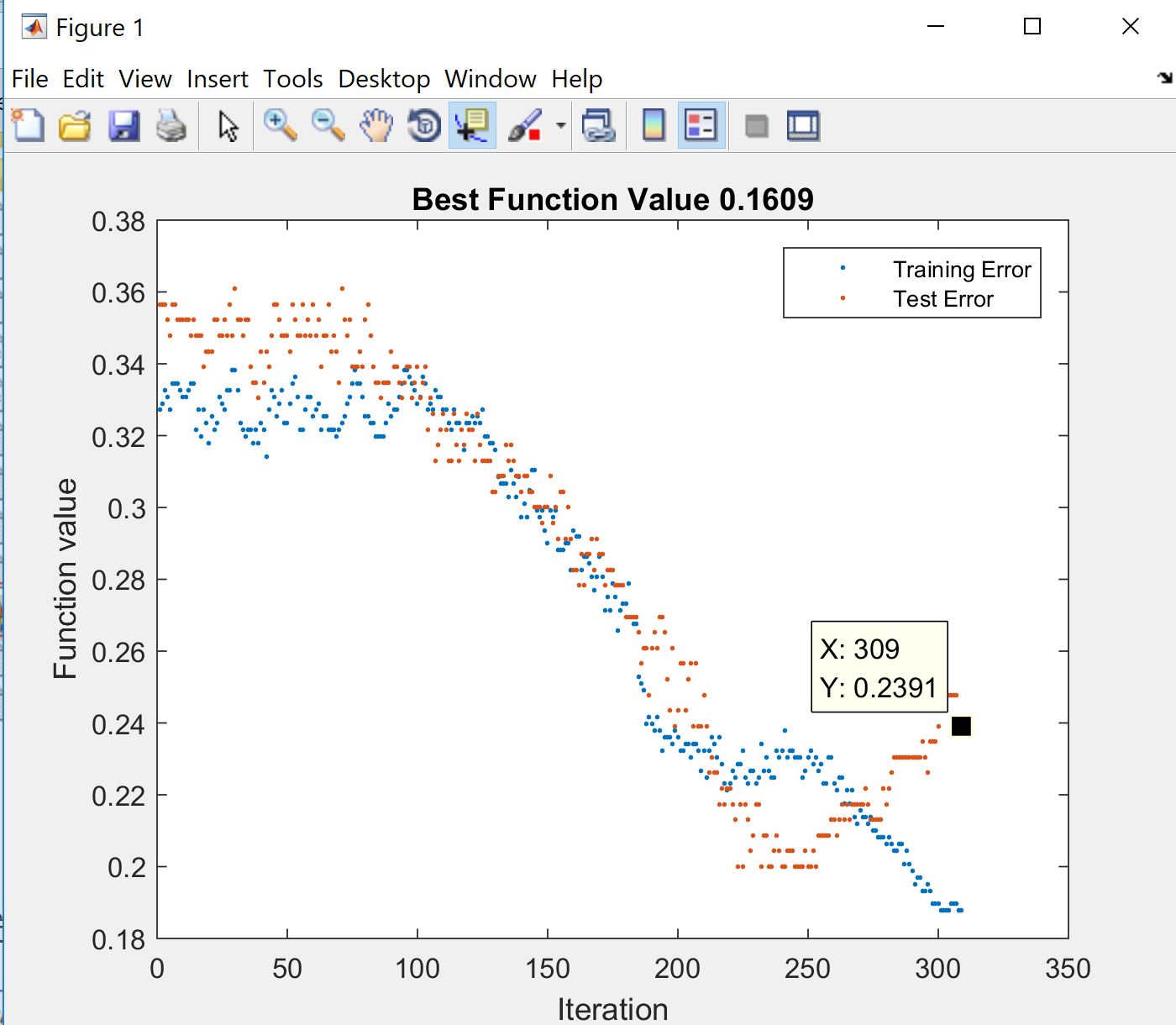
I use the neural networks with default settings, which has 10 hidden layers.

Simulated Annealing:

Result and analysis: Here is the curve for loss function versus number of iterations. The final test error is 0.2391, and the best test error achieved is 0.1609. We can see from the plot that the overall trend for training error is going down, but for test error, it goes down at first, and start to increase. That means that our neural networks start to overfit after 250 iterations of simulated annealing.

Ways to improve: Therefore, we can force the simulated annealing to stop at around 250 iterations (or by adjust the initial temperature and temperature changing rate) for this problem to improve our test performance.

I use function calls to denote the running time for this algorithm, and it cost 70494 function calls to finish the SA.

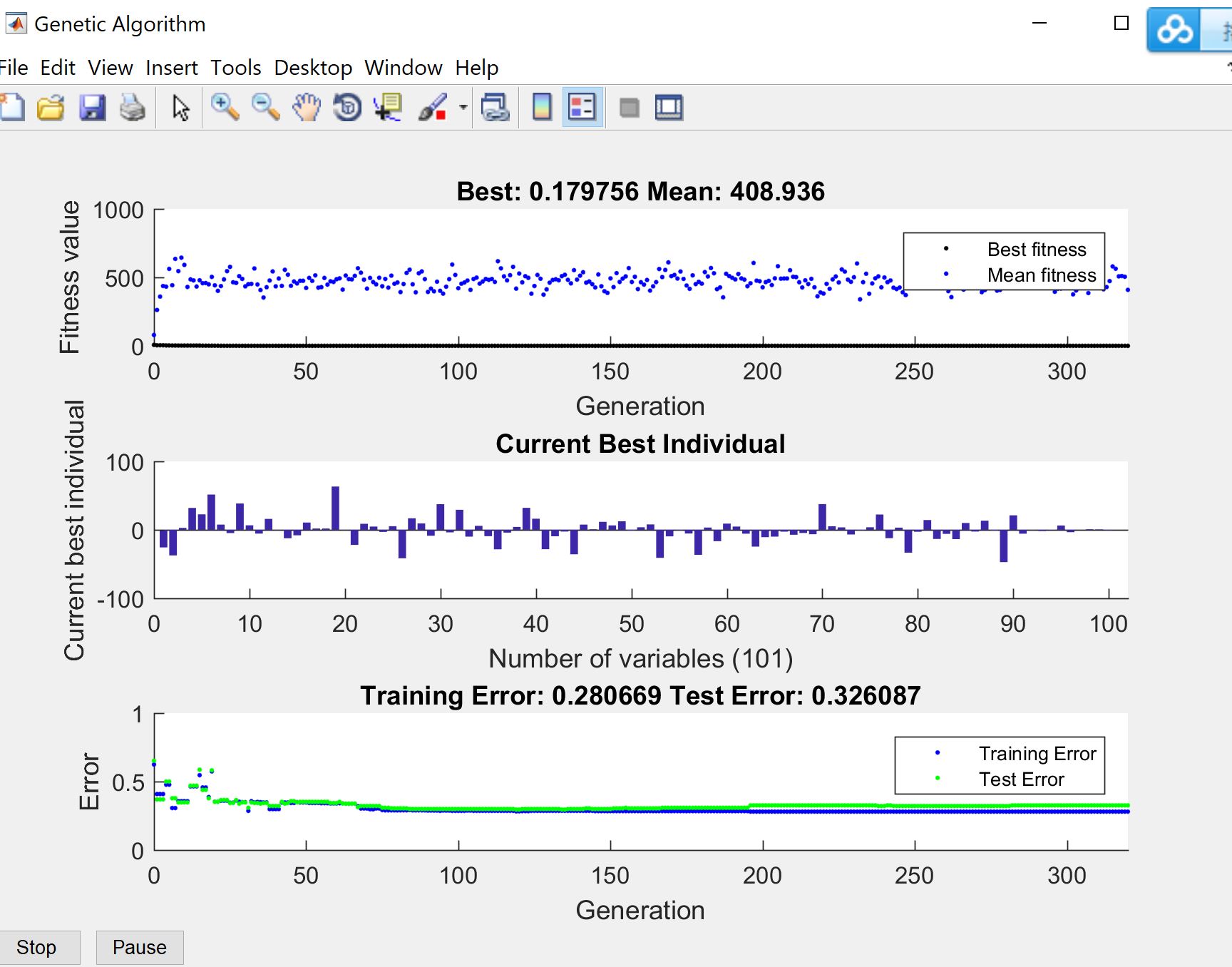


Genetic Algorithm:

Analysis and comparation with SA: Firstly, the training time for GA is much longer than SA. The optimization is terminated due to the exceeding of time limit. Elapsed time is 1209.323692 seconds. It is reasonable because GA utilizes the mutation and cross-over in each iteration, which will require a longer time.

It surprised me that the error of GA is higher than the error of SA, because I thought that GA can explore more than SA because of the mutation and cross-over. I think that maybe in this problem it is better to search around the neighborhood, therefore SA can do better than GA.

Ways to improve: I think that the high error in GA may due to the bad cross-over method. Maybe the cross-over method should be re-designed to improve the result on this problem.

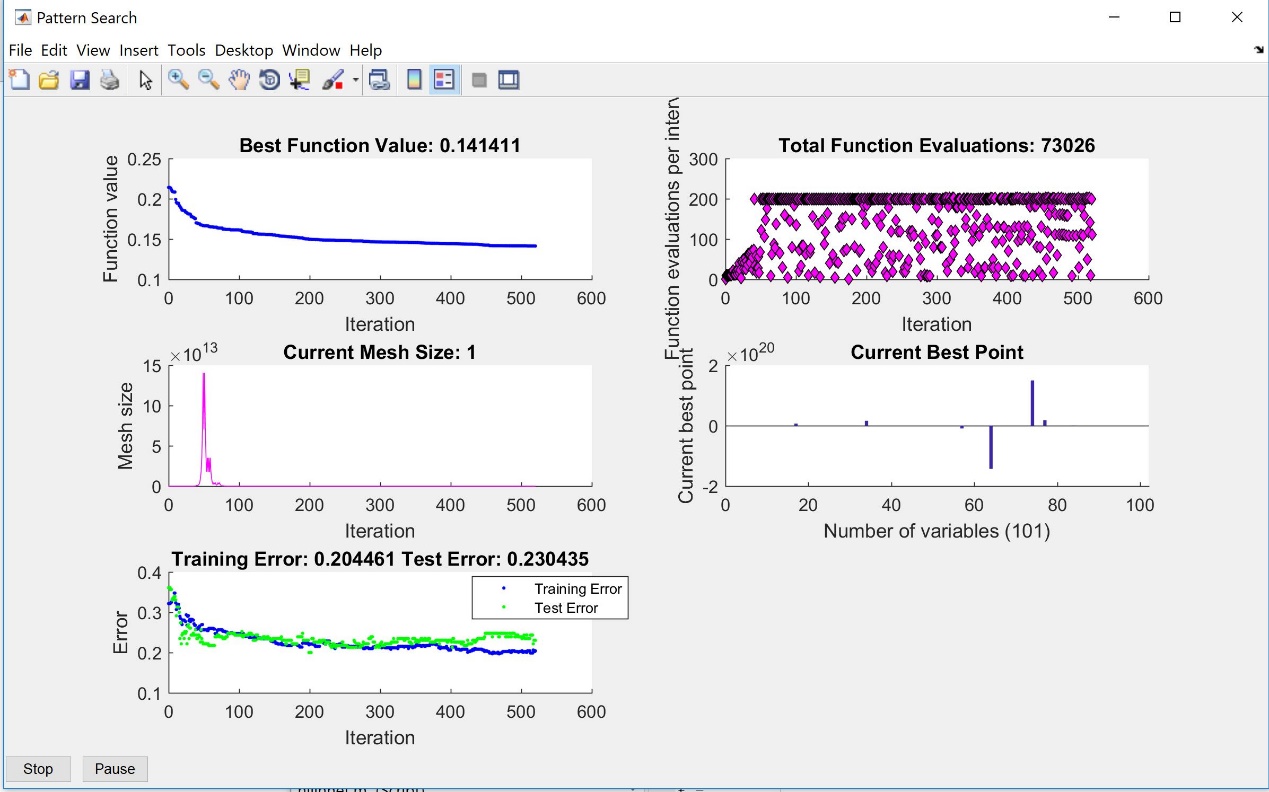


Randomized Hill Climbing:

Analysis and Comparation among the three algorithms: The time required by RHC is the same as the GA, which is around 1206 second. The reason why the RHC took such a long time is because it searched a very large area. I think RHC should take longer time than GA, but it depends on the actual problem.

The test error is around 0.23, which is the best among the three algorithms. I think it is because that the search area is very large, so it is more likely for the RHC to find a more optimal parameter setting for the problem.

Ways to improve: It is hard to improve the training time and maintain the high accuracy because there is a trade-off: The larger you explore (The more time you spend on this problem), the higher chance for you to find the optimal solution.



Conclusion for the Neural Nets Problem:

Here I use a table to summarize what I analyzed in previous sessions. SA requires the shortest clock time, because GA needs to do the cross-over and mutation, RHC needs to search around a large area. SA and RHC have high accuracy for this problem, I think it is mainly because of that the function have many local maxima with high values. And the GA’s bad performance may due to the poorly designed cross-over and mutation method.

|  |  |  |
| --- | --- | --- |
| Algorithms | Clock Time | Accuracy |
| SA | Short | High (in this problem) |
| GA | High | Low (in this problem) |
| RHC | High | High (in this problem) |

**First Problem: Travelling Salesman Problem (Highlight Genetic Algorithm):**

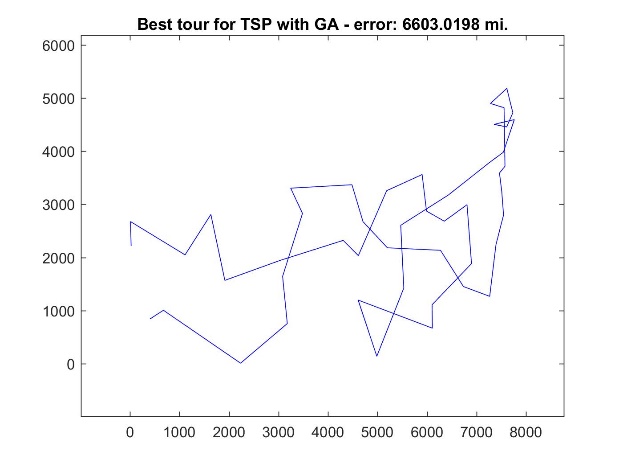
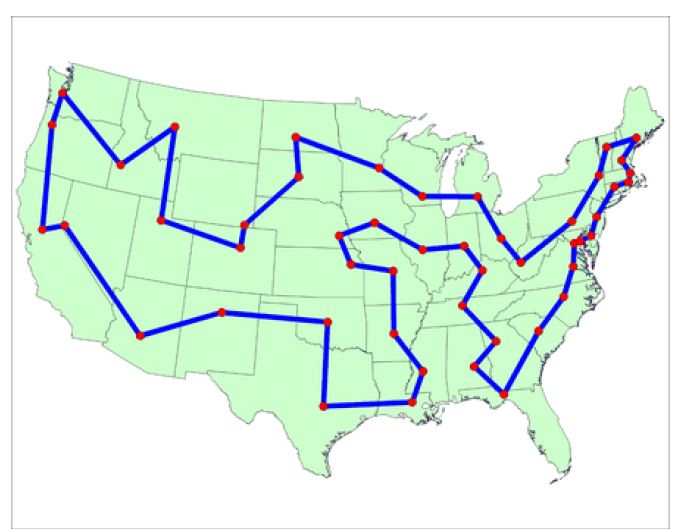
The goal for this problem is to find the path to visit all the cities with lowest cost. Suppose a salesman want to travel around the capitals for 48 states, and the lowest cost is known to be 33523.7 miles. The answer can be found by testing each combination one by one, but that means you need to test 47!/2 = 1.3\* 10^59 routes, which is impossible. Also, the solution will be completely different if there is a small change in parameters, for example, the small change in the location of a city may lead to a completely new solution. Since it is very hard for all the algorithms, the power of genetic algorithms, which utilize the mutation and cross-over to take large steps, will be suitable for this problem.

Result and Analysis:

I run each algorithm 5 times and then calculate the average. The error is the distance longer than the shortest distance. Here I use a table to summarize the result:

|  |  |  |  |
| --- | --- | --- | --- |
| Algorithms | Average Clock Time | Average # of Function Evaluations | Error (meters) |
| SA | 1234 | 22207 | 21705 |
| MIMIC | 83 | 10000 | 32588 |
| GA | 35.4 | 6888 | 6603 |
| Complete poll RHC | 836 | 216 | 16707 |
| First poll RHC | 909 | 4498 | 18261 |

The error for GA is surprisingly low, it is because GA utilizes the mutation and cross-over function to take large steps to explore more possibilities. Luck is a important fact for this problem, and the choice of initial points matters a lot for RHC, SA and MIMIC. However, GA can use mutation and cross-over to get close to the optimal even when it starts at a pool initial condition. I would say that GA is the best approach for this problem, because it took a short clock time and achieve the lowest error. Here are the routes produced by GA (left) and the true optimal routes (right).



SA and two RHCs are at the same level of performance, they took a long clock time and their performance rely largely on the random initial points. Although SA can take some random walks at high temperature, it can not take the steps as large as GA which utilize the mutation and cross-over to reach out of the neighbor. And SA and RHC need to search around a large area around the neighbor, therefore they took a longer clock time.

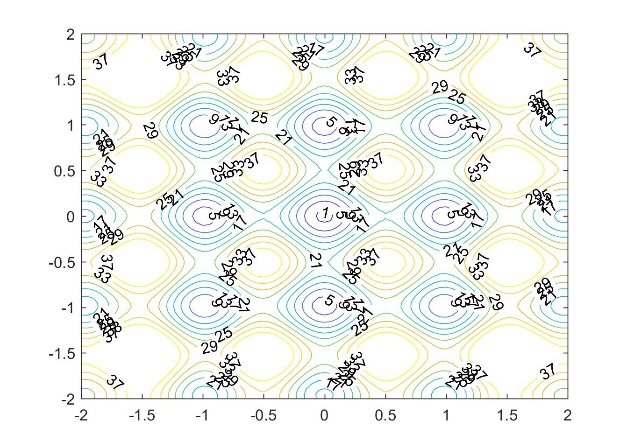
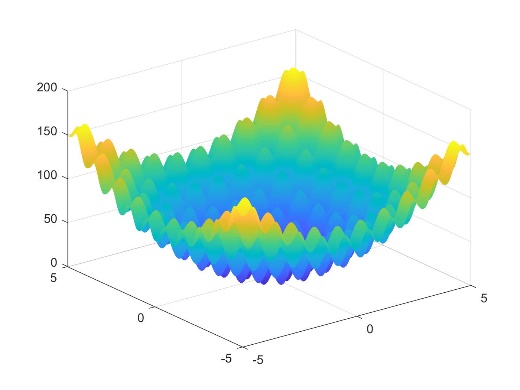
MIMIC did really bad on this problem, it is not surprising at all, because MIMIC’s performance in each iteration will depend on the performance in previous iterations. Therefore, a bad initial point will lead to a failure.

**Second Problem: Rastrigin Function (Highlight Simulated Annealing):**

I found this problem online, and this function is often used for testing the accuracy for genetic algorithms. This function only has one global minimum (x=0, y=0) but have many local minimums, therefore it requires our algorithm to be able to take large steps to explore the points that are not the current best point at each iteration step, to avoid being stuck at the local minimums. Here I use the simplified version of Rastrigin fuction, which have only two variables, x and y. And representing floats (value of x and y) as 32 bit binary vectors permits this function to be treated as a discreet problem.

(𝑥,𝑦)=20+3(𝑥^2+𝑦^2)−10(cos(2𝜋x )+cos(2𝜋x ))

Here are 3D and 2D visualization of this function:



Result and Analysis:

Here we use the distance from the points to true global minimum (x=0, y=0) as our error. And the points that we test are with in this square: {(x , y)| -20<x<20, -20<y<20}. And I run each algorithm for 10 times, then calculate an average time, number of function evaluations and error.

Here I use a table to record the performance of each algorithm on this problem:

|  |  |  |  |
| --- | --- | --- | --- |
| Algorithms | Average Clock Time | Average # of Function Evaluations | Error |
| SA | 0.66 | 12592 | 0 |
| MIMIC | 1700 | 20000 | 1.0 |
| GA | 9.34 | 5977 | 5.79 |
| Complete poll RHC | 2.7107 | 1255 | 0 |
| First poll RHC | 3.08 | 1273 | 0 |

From the table, we can see that **the best algorithm for this problem is SA**, which use the shortest time to achieve almost no error performance for this problem. I think it is because this problem has too many local maximums, the algorithm must be able to explore the neighbor and avoid being stuck at the local maximums. SA can achieve this because it has some random walk feature which is controlled by the temperature.

**In average** GA’s performance is not that satisfying, because **the performance of GA is largely depending on the setting of population**. I changed the population from 5 to 50 during the 10 runs, at beginning, I set the population to be small in order to finish the problem within an acceptable amount of time. However, **it does perform better when the population increases.** **I think maybe the population should be larger for GA to achieve a better performance.**

The normal Random Hill Climbing perform really bad on this problem. And Random Hill Climbing with polling also can explore the neighbor because of the neighbor pooling. However, it takes a longer clock time to do so.

As for MIMIC, the clock time is not that meaningful because it is run in java, the others are powered by MATLAB. And the performance for MIMIC is better than the GA, it is because if we assign high probability to the region near the true global minimum, MIMIC will find the goal precisely. However, due to there are too many “fake” minimum regions, it is also likely for MIMIC to converge to the fake minimum, therefore the error is still high.

**Third Problem: One-Max Problem (Highlight MIMIC):**

This is a very simple problem for human, the goal is just to maximize the number of ones in a bit string. The optimal answer of n-bit string is obvious to us, which is just n. However, the problem is not trivial for the algorithms. Because the value oscillates when you test each combination of bits. And for this problem, the small change in parameters won’t change the result that much (just reduce the number of ones by one or increase the number of ones by one), which is suitable for showing the power of MIMIC.

Result and Analysis:

Here I use 80 bits string for testing, so the optimal value is 80. And I run each algorithm 10 times to calculate the average result in the following table:

|  |  |  |  |
| --- | --- | --- | --- |
| Algorithms | Average Clock Time | Average # of Function Evaluations | Error |
| SA | 4 | 20000 | 35 |
| MIMIC | 15 | 20000 | 3 |
| GA | 0.08 | 60000 | 18 |
| First poll RHC | 0.08 | 50000 | 11 |

From the table we can find that, **in terms of error performance only, MIMIC is the best one.** Because MIMIC can learn and improve the result from the previous iteration, which is suitable for this problem.

And GA and SA are not as good as MIMIC, because they do searching with large randomness, without the guidance from previous iteration. The mutation and cross-over in GA, the random walk in SA will work against the goal in this problem.

As for RHC, it performs better than GA and SA, because it will search around the neighbor to find the best point in the neighbor, however, it resulted in a large search area and more function evaluation.