Stochastic Optimization Algorithms – Home Problem 1

Arunachalam Narasimhan (19940109-7515)

Problem 1.1

1. Initially the function is defined as follows:

1. Now the gradient of the above function is calculated for the given conditions.
2. when the constraints are fulfilled:

1. when the constraints are not fulfilled:

1. For unconstrained condition (), the minimum is found as

The minimum points are

5. The Penalty method is carried out to find the minimum points with the following parameters:

Step size, and Tolerance T = 10^-6

Table 1: Minimum points for various penalty values

|  |  |  |
| --- | --- | --- |
|  |  |  |
| 1 | 0.433777 | 1.210166 |
| 10 | 0.331354 | 0.995540 |
| 100 | 0.313738 | 0.955252 |
| 1000 | 0.311790 | 0.950732 |

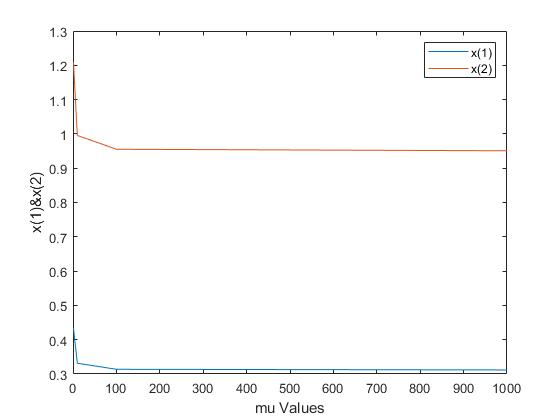


Figure 1:

From the above figure, it is noted that both and values converge to steady state as mu value increases.

Problem 1.3

Here we implement the Genetic algorithm to find the minimum of the function,

1. In this part, the RunSingle.m file is modified with different parameter values and tested for

successful implementation of the algorithm. Finally, a set of parameters were fixed and the

process was repeated 10 times to calculate the following:

Parameters:

Table 1: Parameters for run of 10 epochs

|  |  |  |
| --- | --- | --- |
| populationSize = 100 | numberOfVariables = 2 | crossoverProbability = 0.5 |
| maximumVariableValue = 5 | tournamentSize = 3 | mutationProbability = 0.3 |
| numberOfGenes = 50 | tournamentProbability = 0.75 | numberOfGenerations = 1000 |

With the above parameters, the following values were obtained for 10 runs.

When the simulation was run with more number of generations (2000), it had some room to converge to the minimum point and the points were mostly the same, as opposed to what we see below for a lesser number of generations (1000) where there are variations for ()

Table 2:Function value and Variable value

|  |  |  |  |
| --- | --- | --- | --- |
| **Iterations** | **Function Value** |  |  |
| 1 | 4.39113e-05 | 2.98403 | 0.49569 |
| 2 | 0.000217 | 3.02673 | 0.50444 |
| 3 | 0.000457 | 3.02946 | 0.51087 |
| 4 | 0.000681 | 2.93817 | 0.48316 |
| 5 | 0.000977 | 3.03248 | 0.50210 |
| 6 | 2.7152e-05 | 2.99219 | 0.49893 |
| 7 | 0.000757 | 2.94614 | 0.48984 |
| 8 | 0.00018 | 2.96717 | 0.49150 |
| 9 | 0.000234 | 3.03842 | 0.50880 |
| 10 | 0.003294 | 2.97090 | 0.49561 |

1. Here, the RunBatch.m file was used to run the algorithm for fixed parameter values, except varying the mutation probability ranging from 0 to 1, including 0.02. The process was repeated 100 times for each mutation probability and median and average fitness values were calculated.

Table 3: Fitness values at different Mutation probability

|  |  |
| --- | --- |
| ***Pmut*** | **Median Fitness Values** |
| 0 | 132.464 |
| 0.01 | 6178.126 |
| 0.02 | 75464792.5192 |
| 0.03 | 4983220.0943 |
| 0.04 | 691877.30626 |
| 0.6 | 778.7764 |
| 0.7 | 982.7209 |
| 0.8 | 906.9743 |
| 0.9 | 3440.5597 |
| 1 | 118.262 |

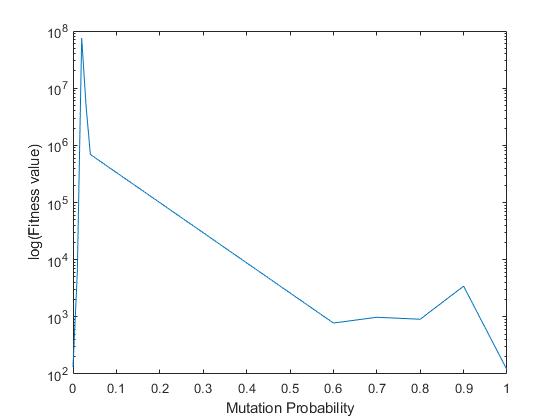
 Below is the plot for the mutation probability Vs median fitness values.

Figure 2: Graph of Pmut Vs Fitness Value

As can be seen from *Table 4*, the algorithm returns the maximum fitness value for mutation probability = 0.02 which corresponds to (1/m), where m is the number of genes (50). This is in accordance to the Onemax function, which finds the binary string for a given length that maximizes the sum of its digits.

1. From the many iterations, we make an initial guess of the actual minimum of () at (3, 0.5), as seen below.

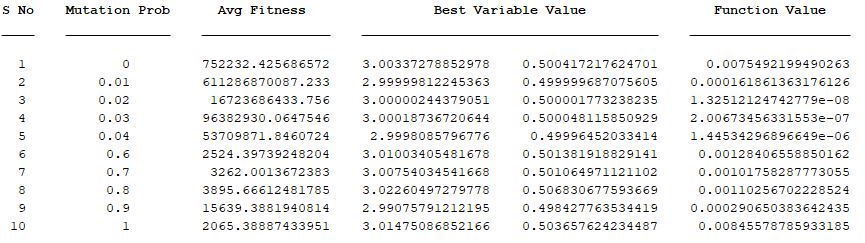
Analytically solving, we should get at (3 , 0.5)

Figure 3: Values after 100 runs of algorithm for each Pmut

substituting (3 , 0.5), we get . From this, it is evident that ( is a stationary point.