Min One problem – A Performance Analysis with Binary-GA

|  |  |
| --- | --- |
| Name: EC Assessment Week 1 | Author: Babu Pallam |
| Date: 23/02/2024 | **Email Id.: P2849288@my365.dmu.ac.uk** |
|  |  |

# Introduction

This report describes the analysis and observations made from an experiment with the given code that solves Min One problem using Binary Genetic Algorithm. The first section discusses the observations found in the execution of the given code and the modifications done to get the median, which is concluded as the iterations needed for reaching the theoretical optimum of this problem. Followed by that, effect of parameters has been discussed; analysis and observation of effect of five parameters, such as Maximum Iterations (MaxIt), Population Size (nPop), Crossover Probability (cP), Mutation Probability (mu), and Beta value (beta), have also been detailed in this section. In the next section, the effect of operators taken into consideration. For which, the operators given in the code for four phases such as, Selection, Crossover, Mutation, and Survivor Selection, have been reimplemented with different operators which are not used in the give code. The performance analysis has been carried out in the same way how it has been done with the given code. In the following section next to this makes a comparison of the result with different operators and different parameter values. Finally, the conclusion has been made.

# Median Calculation

Among several executions made, one solution has been taken arbitrary, which shows 38 iterations required to get the minimum value (which is equal to 0). The cost vs iterations graph with the same result has been provided in Figure 1 in Appendix A with rest of the figures for this section.

 The experiment carried out several times repeatedly subjected to parameters with constant values. During the experiments, a variation in the result, which means the minimum iterations needed to reach the best outcome for the Min One Problem, has been identified. This result varies in between 35 and 45 iterations. To find the median value of this result, results of 30 experiments have been taken into consideration. The code has been modified slightly as it implements 30 runs in a sequence (using a for loop). In addition to that, with the result obtained, a graph has been plotted. The resulting graph is given as Figure 2. The median value of the results of 30 experiments is 40, which is shown in the same figure with a red spot on the axis where y, the output of the fitness function reaches zero. This median value is referred as MedianV1 in the upcoming sections.

Figure 1

The MedianV1 and the values of parameters used, such as MaxIt = 100, nPop = 100, cP = 1, mu = 0, and beta =0.02 to get the result are noted as Benchmark A. The modified code has been provided in Appendix A.

# Effect of parameters

A thorough research has been conducted by analysing and experimenting with the given code by changing values of the parameters. The effects were worth to be noted. The Figures mentioned in this section have been provided in different sections under Appendix B. Note that the given observations are based on the change in parameter values by fixing other values same as that of used in Benchmark A. The main observations are discussed below.

* MaxIt must be of a certain value to let the algorithm converge into the best solution possible, at least it must be greater than the MedianV1 calculated in the above section. The Figure 3 shows the graph of outcomes of different experiments by varying ‘MaxIt’ value range in between 100 and 500. There was no strange pattern observed in analysis while comparing those with the calculated median value.
* Change in nPop also make change in the iteration count to reach the minimum value. The information has been shown in Figure 4. When the nPop value increases, the number of counts also increases but was close enough to the median value. As we can see for the population size 1000, it takes 128 iterations to reach the optimum solution. Moreover, nPop cannot be zero, and for nPop= 1, it takes more than 200 iterations to reach the best result.
* Change in nPop and mu together can make a significant difference in the output. By analysing Figure 4, it can be concluded that as ‘mu’ increases the number of iterations required also increases. Even a small change in ‘mu’ make a huge difference in the output. For instance, mu =0.02 and nPop=100, gives iteration count 41, nearly equal to the MedianV1. When mu = 0.01, Iteration count value has been reduced to 35, and when mu=05, iteration value increased into a value, more than 100. Another point to be noted here is that for different values of nPop, with a constant ‘mu’, the results are almost similar.
* Another observation worth to be noted is the effect of nPop and pC together, which is given in Figure 5. These variables can be considered as primary variables those involved in determining the efficiency of algorithm. As pC increases from 1, the result is reaching to a low value, which is far less than the MedianV1. From mu = 1, it shows an exponential decrease and then it levelled off from 9 to 19 with a value of around 9 after a gradual decrease. The minimum iteration we got is 7 when pC=19 and nPop=500, which has been noted as the best-found value so far.
* Figure 6 has only a small change compared to Figure 5. In Figure 1, mu was 0.02, whereas in Figure 6, mu =0. This has made a change in the output. This time the theoretical minimum has been reduced from 9 to 8. In addition to that, with mu =0, with Y=1, the result has been decreased dramatically for same nPop, while comparing this with the values in Figure 5.
* Beta value is used by selection operator. This value has a significant impact in the tuning of algorithm. As it increases from 1 the change can be seen from the Figure 7. Even the other parameters are at their best tuned value, with beta value equal to 50 or more, the result of Binary Genetic Algorithm has been limited to a particular value as the best solution and keep it constant till end of the execution (see the uniform line parallel to X axis when y not equals to zero). This is where the selection operator fails in this problem.

A conclusion can be made after the performance analysis is that with the given code minimum 7 iterations are needed for the algorithm to reach the optimum solution for the Min One problem. The information of the tuning of the parameter which led to this result has been given below.

* MaxIt with 50 is enough to get optimum solution if other parameters are tuned well.
* nPop cannot be zero, with 100 as population size, it can result one of the best.
* pC effects significantly. pC as 15 to 20 gives best result.
* mu should be less as possible, but with mu=0, there won’t be any mutation, the operation itself will be skipped. So best use the less nonzero value, like 0.001.
* beta gives result best when it is 1.

The result we got here has been considered as Benchmark B. which is used in the following sections often.

# Effect of Operators

The operators chosen in the Min One Problem, solved using binary GA code given and used for determining Benchmark B, are as listed below.

* Selection operation 🡪 Fitness Proportionate Selection Operator
* Crossover operation 🡪 random picking, chose one among three techniques- Single Point, Double Point, and Uniform Crossover
* Mutation operation 🡪 Global Mutation
* Survivor selection 🡪 (Miu+ Lambda) Selection Scheme

For performance testing, different operators have re-implemented in the given code. The operators chosen are listed below.

* Selection operation 🡪 Stochastic Universal Sampling Selection Operator
* Crossover operation 🡪 Binary Three Point Crossover
* Mutation operation 🡪 Local Mutation
* Survivor selection 🡪 (Miu, Lambda) Selection Scheme, given Lambda>Miu (Lambda - size of offspring population and Miu - size of parent population)

Performance analysis has been carried out in a similar way how benchmark B has been determined. The information related to the experiment done is provided in Appendix C under different sections such as graphs and MATLAB codes. After several successful experiments with the new code, the median value has been calculated and it is found to be 201.5, which is referred to as MedianV2 in this report. Figure 10 provides the graph. The main observations taken from the experiments done are discussed below.

* The performance of algorithms has shown no significant change in terms of iteration count in producing the best value on different iteration count set keeping rest values as of Benchmark A. The results were close to the MedianV2. A well-defined illustration has been provided in Figure 11. As the graph provides, the results of the experiments done with MaxIt=100 to MaxIt=500 are all close to the MedianV2.
* nPop reflects the output significantly. As per the analysis given in Figure 12, for lower values of nPop, such as 50, and 100, it takes more than 100 iterations, whereas for value 300 there is a significant reduction can be observed in the iteration count to 68. Further, higher values like 1000 and 5000 the iterate count found to be 53 and 28 respectively.
* cP and nPop together make a big difference in the iteration count in order to get best result. As can be seen in Figure 13, the result varies from 27 to 101. For different values of nPop with cP = 1, the iteration count is near to 100. It is evident to see that, for the same nPop, the iteration count is high for lower values of cP and low for higher values of cP. Further, for same cP, as nPop increases the iteration count has seen a gradual reduction. The best lowest value which has been found during this experiment is when nPop = 400 and cP=17, which resulted in 27, which is far best than the MedianV2.
* Like cP and nPop, cP and nPop also make an impact on the performance of algorithm. Figure 14 provides a detailed view. Unlike other graphs, this graph shows a strange pattern. For lower values of mu and nPop, the iteration count is high. Furthermore, a linear increase of either both the values or one value while keeping others constant, the iteration count decreases. The upper bounds of the experiment, nPop= 5000 and mu=1, the result is 21, which is less than the value we got in the above observation. The best iteration count for the optimum value has been found out to be 20, for the values nPop= 4000 and mu=1.
* In case beta value, the effect is similar as what have been seen from Benchmark B. the first observation is that the algorithm fails when beta is greater than 3, since no value can be indexed while applying selection operator since probability of selection of each solution is nearly equals to zero. For negative values of beta, the algorithm results to best solution which is close near to MedianV2.

The best result settled after the performance analysis of modified code is that minimum 20 iterations are needed for the algorithm to reach the optimum solution for the Min One problem. The values and result formulated in this section is referred to as Benchmark C, which is used in the following section. The conclusion has been made after the analysis of the performance results, which shows the minimum iteration count can be found by tuning the parameter in the following manner.

* MaxIt set to 250 is a good choice since it is above the median value.
* nPop should be high if the iteration count needed to be reduced, greater than 1000 is advisable.
* cP should also need to be high (near to 20 is advisable).
* mu should be set a high value (advisable value is 1) for the best result.
* Keeping beta as 1 or less than that is good enough since it makes a significant impact on the selection process.

# Comparison

The Benchmark A, B and C have been compared in terms of performance. The following conclusions are drawn from observation and analysis.

* The MedianV2, from Benchmark C, has been found very high while comparing with MedianV1, which is found in Benchmark A, with the same parameter values used in Benchmark A. The solution made with Benchmark A is more efficient in terms of best number of iterations needed to get optimum solution for this Min One problem with parameter values keep to its lower values.
* For a best outcome, with the cost used in Benchmark B, the problem can reach to its optimum minimum with 7 iterations when MaxIt= 50, nPop= 500, cP= 19, mu= 0, and beta=1. For the code used in Benchmark C, the best outcome can be attained in 18 iterations when MaxIt= 50; nPop= 1000; cP= 20; mu= 1; and beta= 1. The results obtained in the experiment done have been provided as Figure 16 and Figure 17 in the Appendix D.
* After comparison of Benchmark B and Benchmark C, an explanation can be drawn in the reduction of number of iterations in both Benchmark B and C, when nPop and cP are higher. When we have these two values higher, the population space that have been selected by the selection operator in each iteration is big that is enough to comprise several solutions. So, the probability of inclusion of the solution with best possible outcome (fitness value) is higher.

# Conclusion

TheMin One problem with the given code has run smoothly and has taken into consideration for further analysis. The code run for several times and find the minimum iterations needed to reach the theoretical minimum. The median value of that has also been calculated, which was found to be 40, referred to as MedianV1. The effect of parameters has been identified, analysed, and concluded with the given code; how the parameters should be tuned to get best solution with minimum amount of time has been illustrated with graphs. In addition to that, the effect of operators used in the algorithm has been examined by reimplementing the problem with operators which are different from what have been used in the given code for the four phases of EA, such as selection, crossover, mutation, and survivor selection. Finally, a comparison of the performance with the modified code against the given code has also been discussed. In addition to that, the reasons for why we get the best result while changing the parameter values has also been explained.

# Appendix A

## A1. Finding Median of Results (Benchmark A)



Figure 2

### MATLAB Code:



# Appendix B

Performance Analysis Results with graphs and the Modified Codes (BENCHMARK B)

## B1. Effect of Parameter: Maximum Iteration (MaxIt)

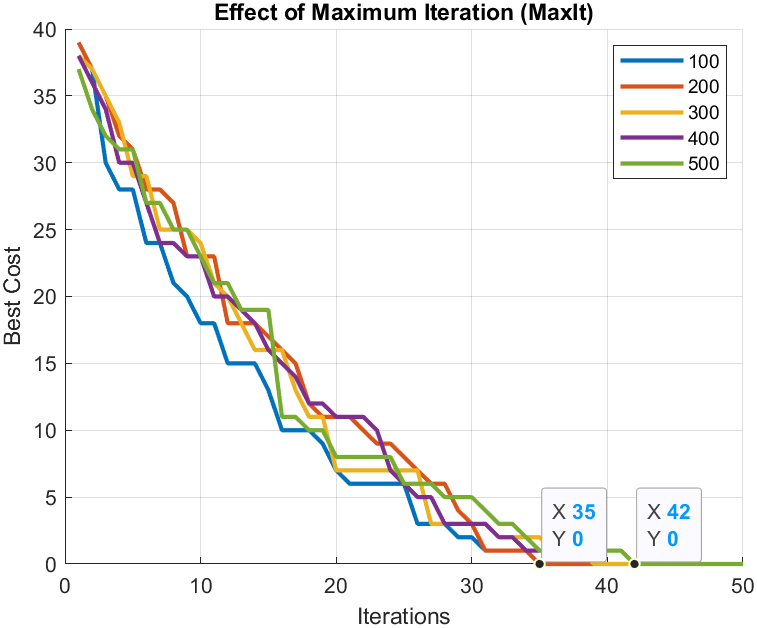


Figure 3

### MATLAB Code:



## B2. Effect of Parameter: Population Size(nPop)

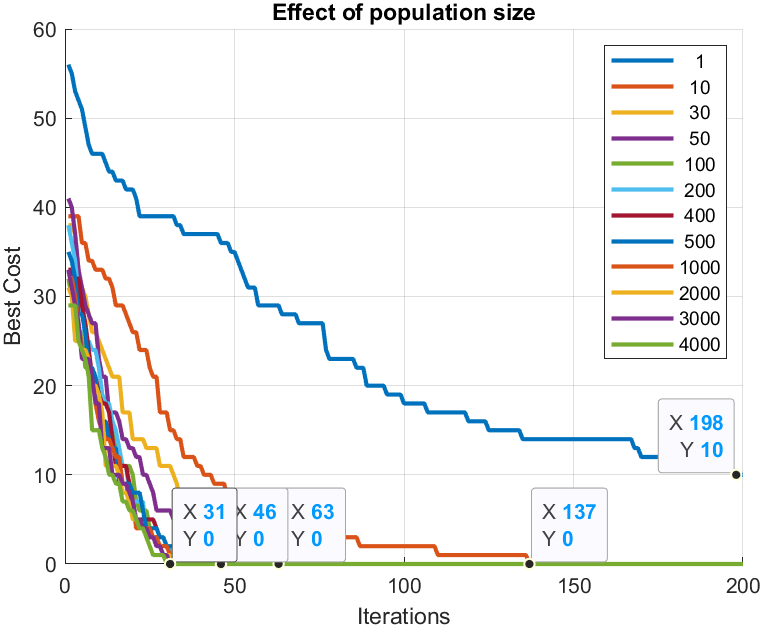


Figure 4

### MATLAB Code:

****

## B3. Effect of Parameter: Population Size(nPop) & Mutation Probability(mu)

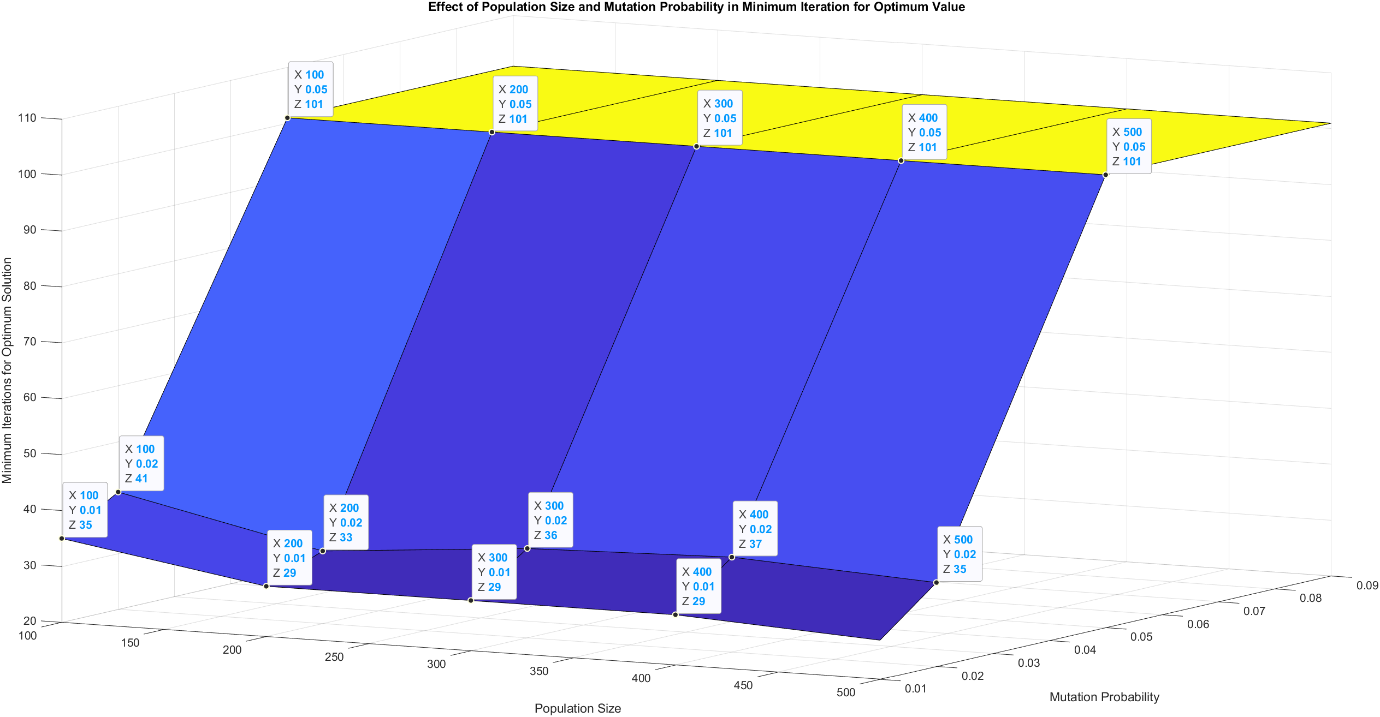


Figure 5

### MATLAB Code:



## B4. Effect of Parameter: Population Size(nPop) & Crossover Probability(pC)

A diagram of a graph

Description automatically generated

Figure 6

### MATLAB Code:



## B5. Effect of Parameter: Population Size(nPop) & Crossover Probability(pC) with mu = 0

A graph of a graph of a graph

Description automatically generated with medium confidence

Figure 7

### MATLAB Code:



## B6. Effect of Parameter: Beta value(beta)

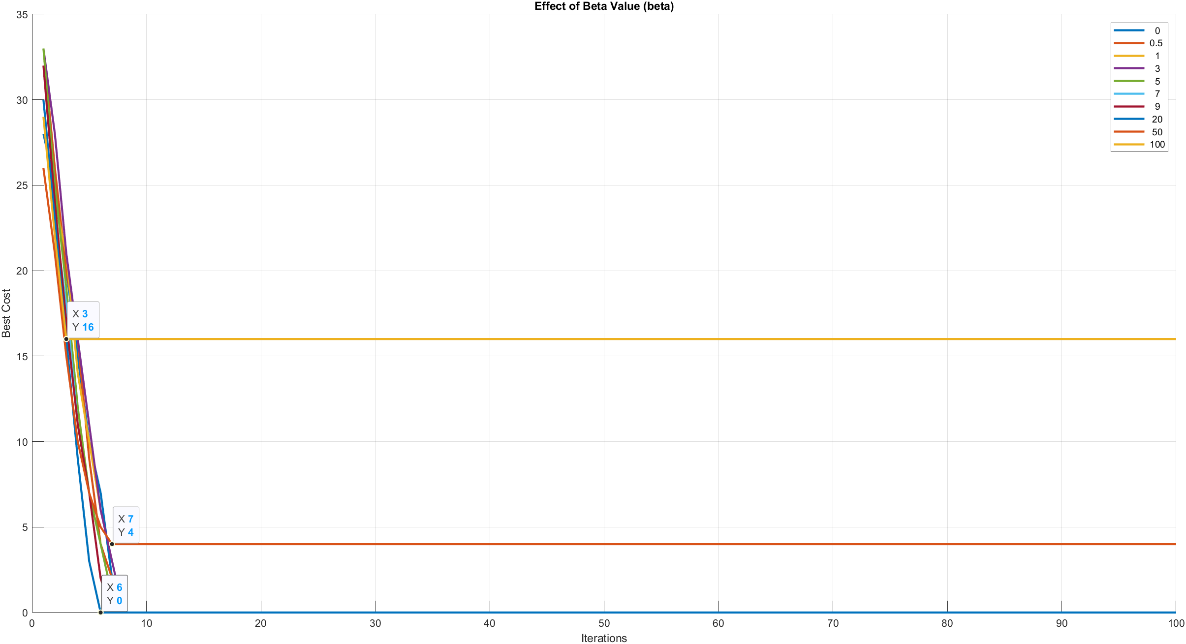


Figure 8

### **MATLAB** Code:



# Appendix C

Performance Analysis Results with graphs and the Modified Codes (BENCHMARK C)

## C1. Codes that Implement the Operators

### Stochastic Universal Sampling Selection Operator



### Crossover operation 🡪 Binary Three Point Crossover



### Mutation operation 🡪 Local Mutation



### Survivor selection 🡪 (Miu, Lambda) Selection Scheme



## C2. A successful Run After the Reimplementation of Operators

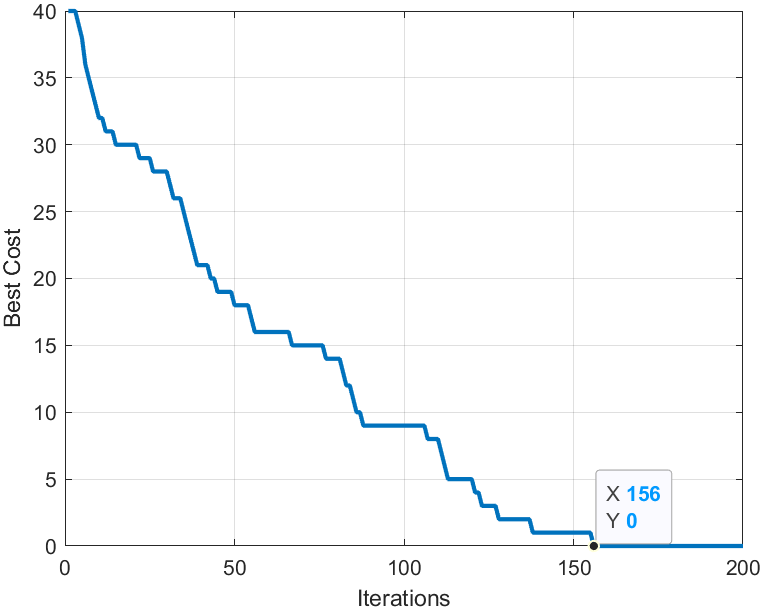


Figure 9

### MATLAB Code:



## C3. Finding Median of Results (MedianV2)

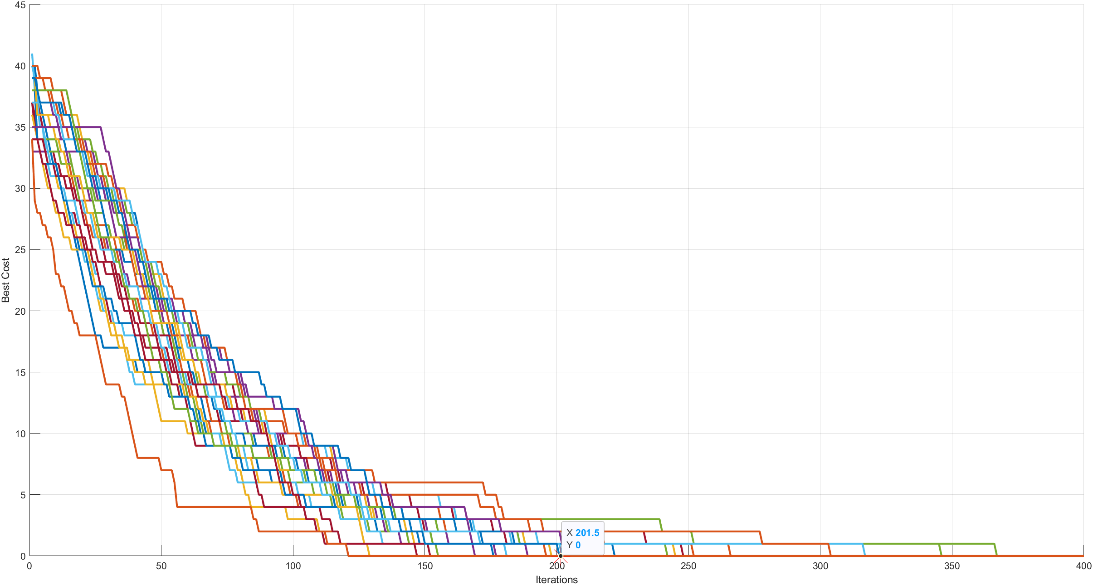


Figure 10

### MATLAB Code:

****

## C4. Effect of Parameter: Maximum Iteration (MaxIt)

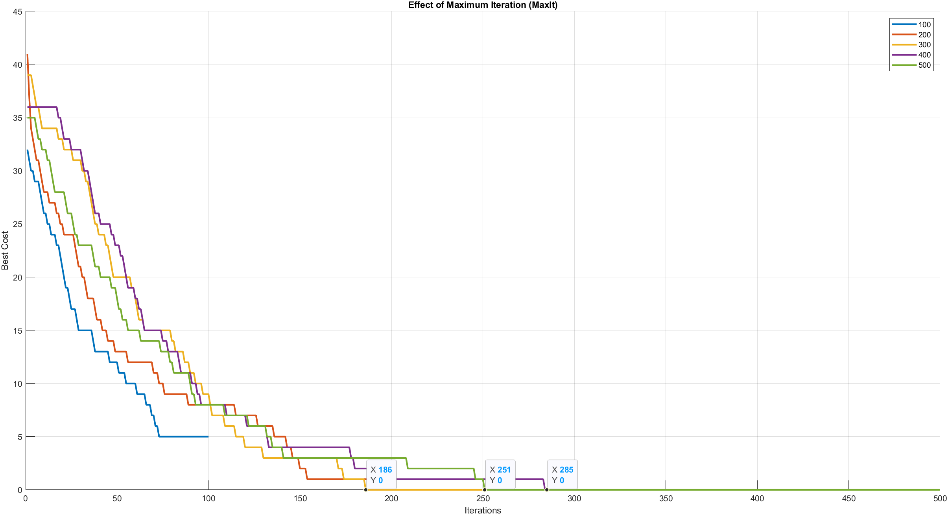


Figure 11

### MATLAB Code:



## C5. Effect of Parameter: Population Size(nPop)

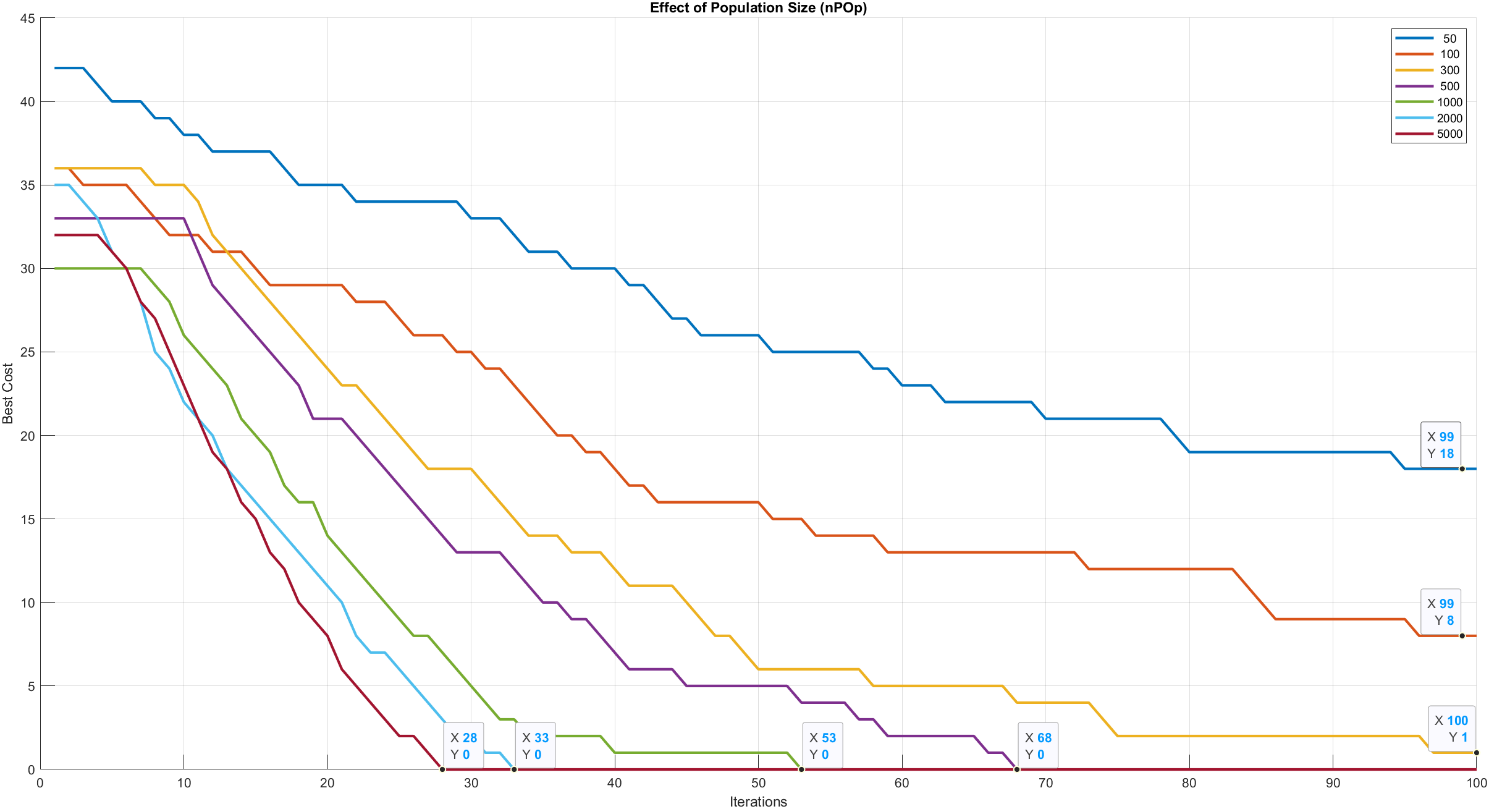


Figure 12

### MATLAB Code:



## C6. Effect of Parameter: Population Size(nPop) & Crossover Probability(pC)

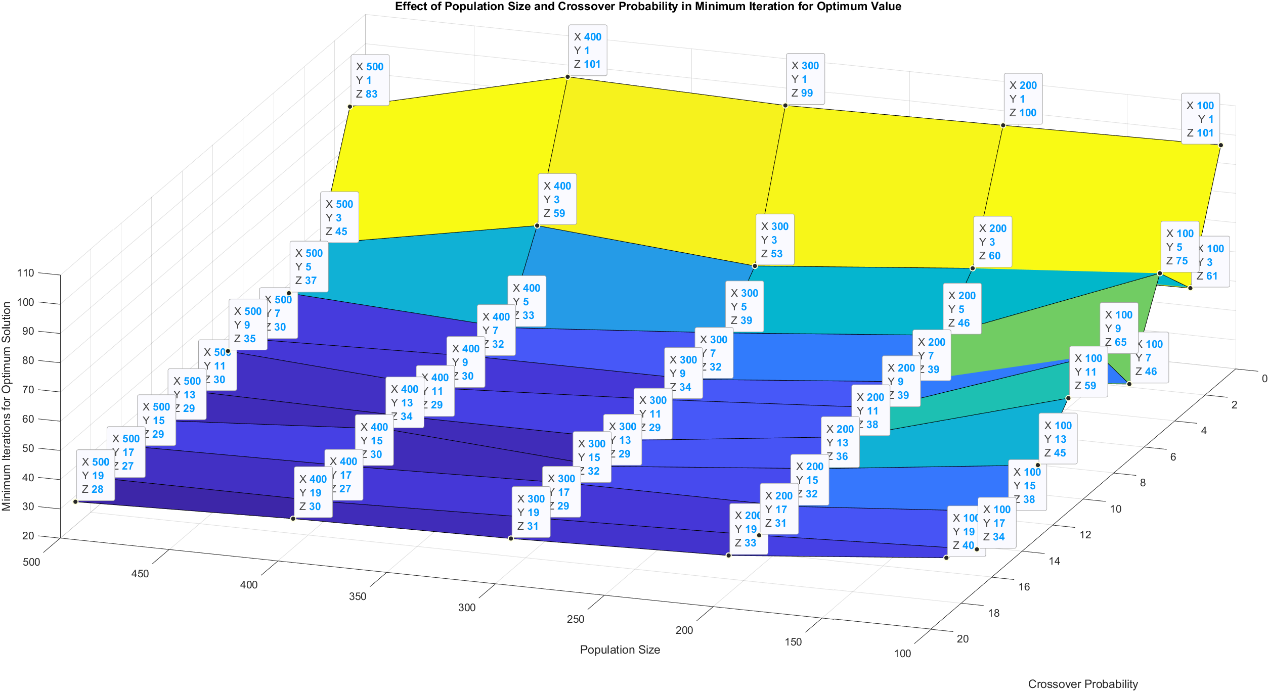


Figure 13

### MATLAB Code:



## C7. Effect of Parameter: Population Size(nPop) & Mutation Probability(mu)

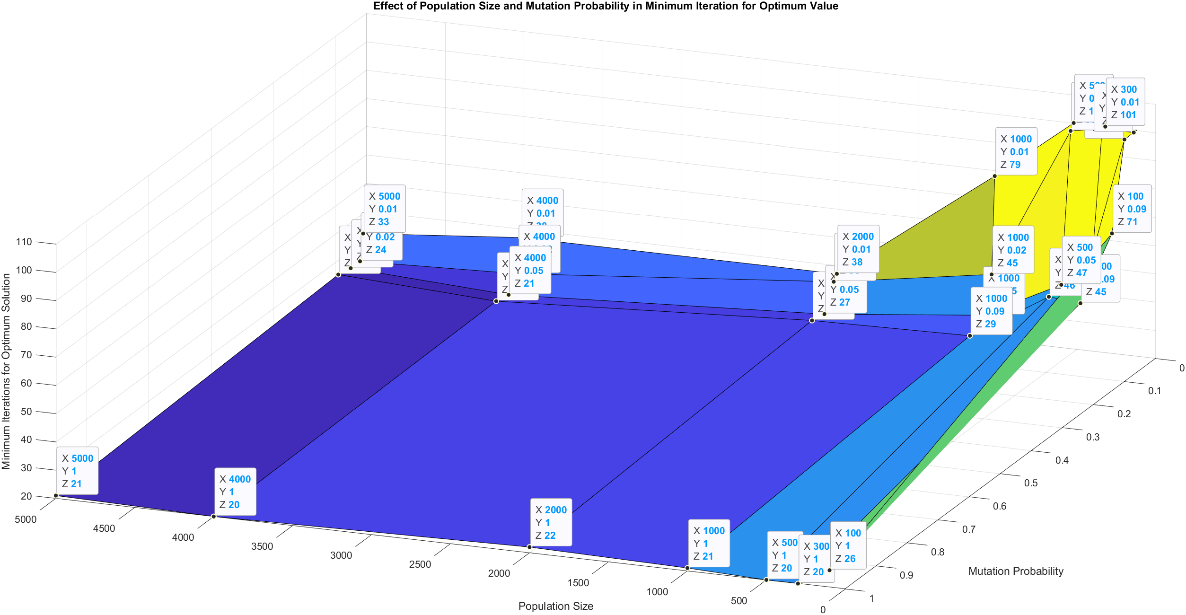


Figure 14

### **MATLAB** Code:



## C8. Effect of Parameter: Beta value(beta)

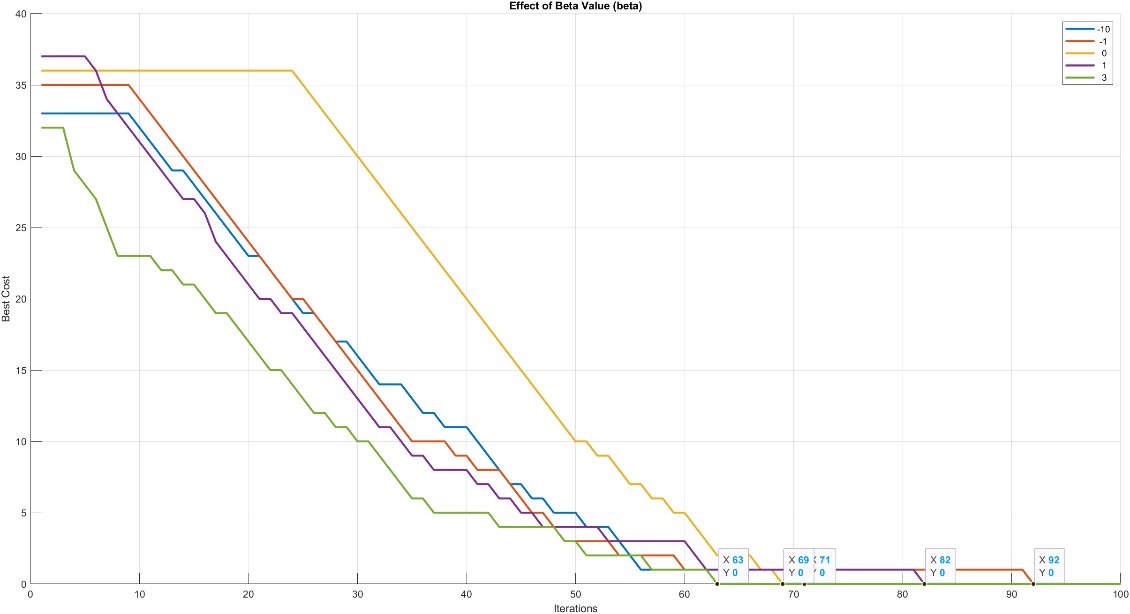


Figure 15

### **MATLAB** Code:



# Appendix D

## D1. Best Outcome with Benchmark B

A graph with a line

Description automatically generated

### Part of MATLAB Code for Tuning:



## D2. Best Outcome with Benchmark C

A graph with a line

Description automatically generated

### Part of MATLAB Code for Tuning:

