

OHM Project (IM39003)

Genetic Algorithm based model for Optimizing Bank Lending Decisions

Divayum Gupta 17IM30010

Dept. of Industrial and Systems Engineering

IIT Kharagpur



Contents

Introduction to the Problem Statement.....	3
Benefits of this model.....	3
Methodology	4
Decision Variables.....	5
Chromosome Encoding.....	6
Validation of a possible solution to determine feasibility	7
Fitness Function.....	8
Loan Revenue (ϑ)	8
Loans Cost (μ).....	8
Total Transaction Cost (ϖ)	8
Cost of Demand deposit (β).....	8
The Algorithm.....	9
1) Initializing population.....	9
2) Evaluating fitness.....	10
3) Selection	10
4) Crossover	10
5) Mutation.....	11
6) Checking Validity.....	11
6) Termination	11
Visualising Results	12
Comparison with another optimization model (Simulated Annealing).....	13
References.....	14
Annexure.....	14

Introduction to the Problem Statement

In times of financial crisis, illiquidity becomes a major issue for both lenders as well as borrowers. Insolvency of banks can lead to a reduction in bank loans. This is due to two reasons. Namely:

- 1) Banks with inefficient allocation of loans have a large proportion of loans which are at risk of default reducing the capital available to these banks [1]
- 2) Moreover, even healthy banks which have more efficient allocation of loans become more cautious in their lending practices through cutting back on lending [1]

Due to these reasons, there is a credit crunch which makes it essential to develop an optimal mechanism for determining bank lending decisions which maximize the bank profits in a timely manner.

The issue at hand here is not whether the borrowers are eligible to get the required loan but rather, to make lending decisions in a credit crunch environment where all applicable borrowers are eligible to get the desired loans. This, therefore, is an NP-hard optimization problem which can be solved using meta-heuristic algorithms such as evolutionary algorithms [2]

The paper [3] therefore proposes a model based on Genetic Algorithm (GA) to organize bank lending decision in a highly competing environment with credit crunch constraint.

Benefits of this model

This GA model allows banks to make efficient lending decisions when faced with a liquidity shock. The focus of the model is two-fold:

- 1) To stabilize systematically banks while achieving maximum profit
- 2) Establish the capital base so that banks can increase lending efficiently

The objectives of the model are:

- 1) Determine a bank lending decision that maximizes bank profit
- 2) Determine a bank lending decision that minimizes the crediting cost

This model takes into account the margins along which banks adjust their loan portfolios in response to the crisis.

Methodology

A basic assumption for this model is that all borrowers which are considered are eligible for the required loans. The model is used to search for the best selection of loans depending on borrower factors such as:

- 1) Loan Age
- 2) Loan Size
- 3) Loan Type
- 4) Credit Rating
- 5) Credit Limit

Moreover, bank variables that are considered for selecting this optimal solution are:

- 1) Loan Interest Rate
- 2) Expected Loan Loss
- 3) Deposit Rate
- 4) Reserve Ratio
- 5) Transaction Cost

The flow of information in the model can be better understood with Figure 1.

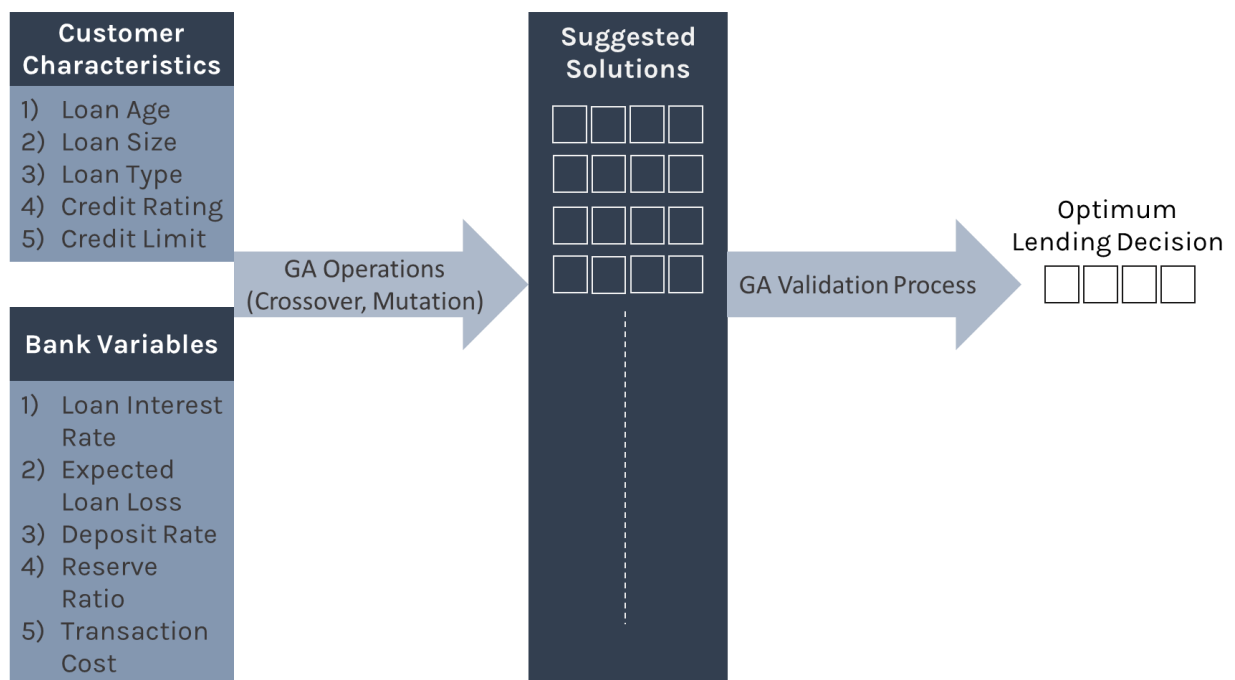


Figure 1 – Working steps of GAMCC

Decision Variables

1) Loan Age (α)

It varies between 1 and 20 years

Category (α)	Value
1	$1 \leq \alpha \leq 3$
2	$3 < \alpha \leq 5$
3	$5 < \alpha \leq 10$
4	$10 < \alpha \leq 20$

Figure 2 – Loan Age Categories

2) Credit Limit

This is the maximum loan amount a borrower is eligible for.

3) Loan Size (L)

It is the amount of loan requested by the borrower and is an important variable as it affects the total allowable loan amount of the bank

Category (L)	Value
Micro	$\$ 0 \leq L \leq \$ 13,000$
Small	$\$ 13,001 < \alpha \leq \$ 50,000$
Medium	$\$ 50,001 < \alpha \leq \$ 100,000$
Large	$\$ 100,001 < \alpha \leq \$ 250,000$

Figure 3 – Loan Size Categories

4) Loan Type (ϕ)

There are three types of loan type in the model:

- a) Mortgage (M)
- b) Personal (P)
- c) Auto (A)

For both personal and auto loans it has been assumed that loan age is less than 10 years.

5) Loan Interest Rate (rL)

Interest rates are determined using loan age and loan type. These are defined for each borrower based on the mentioned parameters.

6) Expected Loan Loss (λ)

Borrower credit rating is used to determine the range of the expected loan loss. The borrowers are divided into categories on the basis of their loan loss.

Credit Rating	λ Value
AAA	$0.0002 \leq \lambda \leq 0.0003$
AA	$0.0003 < \lambda \leq 0.001$
A	$0.001 < \lambda \leq 0.0024$
BBB	$0.0024 < \lambda \leq 0.0058$
BB	$0.0058 < \lambda \leq 0.0119$

Figure 4 – Credit ratings and expected loan loss ranges

Chromosome Encoding

The genes in the chromosome have binary encoding (and therefore have values 0 or 1). The length of the chromosome is equal to the number of eligible borrowers. Each borrower is therefore represented by a gene in the chromosome. A 0 in the gene means that the loan was not provided to the borrower. Similarly, a 1 in the gene means that the loan was provided to the borrower. Each customer is identified by their index number in the chromosome. Therefore, each chromosome with combinations of 0s and 1s represent a loan allocation solution. An example for a chromosome for a dataset consisting of 10 borrowers is shown in Figure 5.

Figure 6 shows a sample dataset of 10 borrowers for which Figure 5 can be considered a possible solution (possible, **not necessarily feasible**)

1	2	3	4	5	6	7	8	9	10
0	1	1	0	0	1	1	1	0	1

Figure 5 – A chromosome consisting of 10 genes corresponding to 10 borrowers

	Loan Age	Loan Size	Loan Type	Credit Rating	Interest Rate	Loan Loss
0	10	79959.32	M	AAA	0.021	0.0002
1	25	64334.01	M	BB	0.022	0.0058
2	4	90327.01	M	A	0.021	0.0001
3	11	94408.92	M	AA	0.027	0.0003
4	18	84621.19	M	BBB	0.025	0.0024
5	3	71544.00	M	AAA	0.026	0.0002
6	17	90366.16	M	BB	0.023	0.0058
7	15	81913.15	M	AAA	0.021	0.0002
8	9	67991.48	M	A	0.028	0.0010
9	10	55615.93	M	A	0.022	0.0010

Figure 6 – A sample dataset of 10 borrowers

Validation of a possible solution to determine feasibility

When generating a solution, we must ensure that it is feasible and within our constraints of total allowable loan amount of the bank.

Essentially, this can be done by calculating the total amount of loan allocation for a solution and comparing it with the total allowable loan amount of the bank.

The value of the total allowable loan amount of the bank is given by the formula $(1-K)D$.

ID	1	2	3	4	5
Gene	0	1	1	0	0
Loan Amount	\$79,959.32	\$64,334.01	\$90,327.01	\$94,408.92	\$84,621.19
ID	6	7	8	9	10
Gene	1	1	1	0	1
Loan Amount	\$71,544.00	\$90,366.16	\$81,913.15	\$67,991.48	\$55,615.93

Figure 7 (a) – An example of Chromosome vs Loan Amounts for each gene

$$\sum L = 64,334.01 + 90,327.01 + 71,544.00 + 90,366.16 + 81,913.15 + 55,615.93$$

If $\sum L \leq (1 - K)D$, solution is valid

Figure 7 (b) – Calculating Total Loan Amount and Validating solution

Thus, if the total loan amount is less than the total allowable loan amount of the bank, the solution is feasible.

Fitness Function

The fitness function (F_x) consists of the following components:

- **Loan Revenue (ϑ)**

The value of the loan revenue is calculated using the loan interest rate (r_L), loan size (L), and the expected loan loss (λ).

$$\vartheta = \sum_{i=1}^n (r_L L - \lambda)$$

The summation is over all the genes with value 1.

- **Loans Cost (μ)**

The value of the loan cost is determined using the loan size (L) and the predetermined institutional cost (δ).

$$\mu = \sum_{i=1}^n L\delta$$

The summation is over all the genes with value 1.

- **Total Transaction Cost (ϖ)**

The value of the total transaction cost is determined using institute transactional cost (T) and the customer transaction rate (r_T). The value of r_T has been assumed to be 0.01 for the purpose of this project.

$$\varpi = \sum_{i=1}^n r_T T$$

Here, the institute transactional cost (T) is determined using the below formula:

$$T = (1-K)D$$

The summation is over all the genes with value 1

- **Cost of Demand deposit (β)**

The value is determined using the bank's deposit interest rate (r_D) and the bank's deposit (D)

$$\beta = r_D D$$

Finally, the fitness function F_x is given by the below formula: **(refer annexure)**

$$F_x = \vartheta + \mu + \varpi - \beta - \sum_{i=1}^n \lambda$$

The Algorithm

The genetic algorithm can be represented through the flowchart as shown in Figure 8.

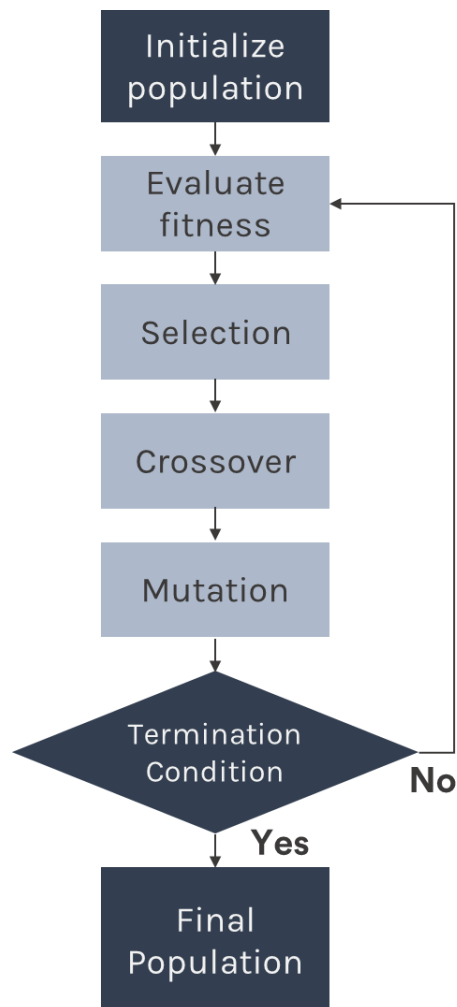


Figure 8 - Genetic Algorithm flowchart

The steps are as below:

1) Initializing population

```
ga_generate_init_pop(df, N, n, K, D)
```

We randomly generate an initial population of chromosomes and validate each randomly generated chromosome (rgc) for feasibility to create a population of feasible solutions. The function returns a list of chromosomes which are represented as a list. The length of chromosomes is the size of the dataset, and the length of the list is the population size.

2) Evaluating fitness

```
ga_fitness_eval(population, N, df, IC, K, D, rD, rT)
```

We evaluate the fitness of each chromosome in the population and add it to the fitness list. On the basis of this we are able to determine average fitness of a generation and the maximum fitness of each generation. These are plotted along with generations to visualize the progress of the genetic algorithm. Also, fitness values affect the probability of selection of a chromosome for reproduction meaning that chromosomes which have higher fitness values are more likely to be selected in future generations and pass on their genes. This function returns a list of fitness values. Also, in order to easily calculate probabilities, it has been ensured that all the values in the fitness are positive.

3) Selection

```
ga_selection(population, fitness)
```

We select our mating pool from the original population on the basis of fitness values. The probability of selection of each chromosome is proportional to its corresponding fitness value. More precisely, the probability (P) of selection of each chromosome (C) is given as:

$$P = \frac{F_x(C)}{\sum_{i=1}^n F_x(C_i)}$$

We create a mating pool of the same size as the original population consisting of chromosomes selected with the previously described probabilities. The function returns a list of chromosomes. The length of the chromosome is unchanged, and the length of the list is the population size.

4) Crossover

```
ga_crossover_chromosomes(selected, crossover_ratio, K, D, df)
```

In the crossover function we create pairs of chromosomes, then on the basis of the probability of a crossover occurring called the crossover ratio, we randomly determine which pairs will undergo crossover. For the pairs which undergo crossover, we determine the position of crossover. Here we conduct a one-point crossover. We must ensure that the position of crossover is not an end gene in the chromosome otherwise it would mean no crossover is occurring. Before appending the chromosome to the population, we validate it. The crossover can be better represented as in Figure 9. Function output of same dimensions as input.

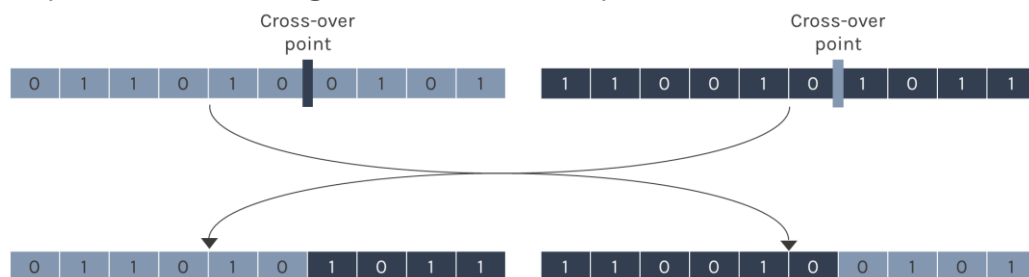


Figure 9 – Crossover of chromosomes

5) Mutation

```
ga_mutation(crossover_result, mutation_ratio, K, D, df)
```

Mutation is the process of randomly inverting the values of genes in the chromosome to create variety in the solutions. We randomly determine the chromosomes to carry out the mutation on, with the probability equal to the mutation ratio. For the chromosomes where the mutation is to be carried out, a random gene is selected, and its value is inverted. We then validate the mutated chromosome and only then append it to the mutated population. This can be better understood from Figure 10.

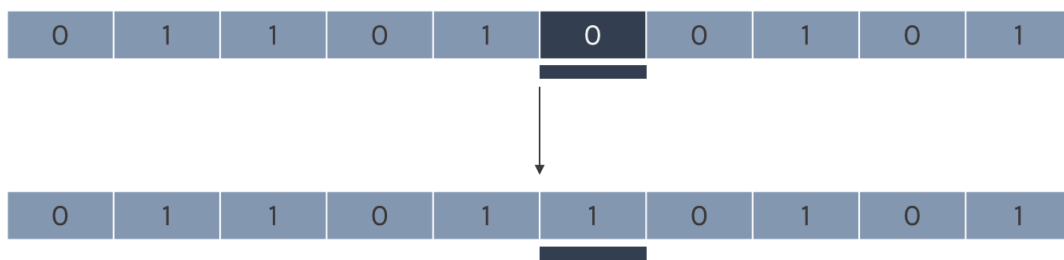


Figure 10 – Mutation of a chromosome

6) Checking Validity

```
ga_check_validity(chromosomes, K, D, df)
```

At every step of generating a new chromosome, we have to be careful to maintain the solutions within the constraints. Therefore, we need to validate the feasibility of the solutions using the check validity function. It is very basic and simply returns a 0 for an invalid solution and 1 for a valid solution.

6) Termination

```
for i in range(gen-1):  
    ...
```

When the termination condition is reached – the required number of generations created or no significant change in average fitness over multiple generations – the final population is arrived at and returned.

Visualising Results

We maintain a history of average fitness values and maximum fitness values for each generation of the algorithm. These allow us to visualize the progress of the algorithm in reaching the near-optimal solution.

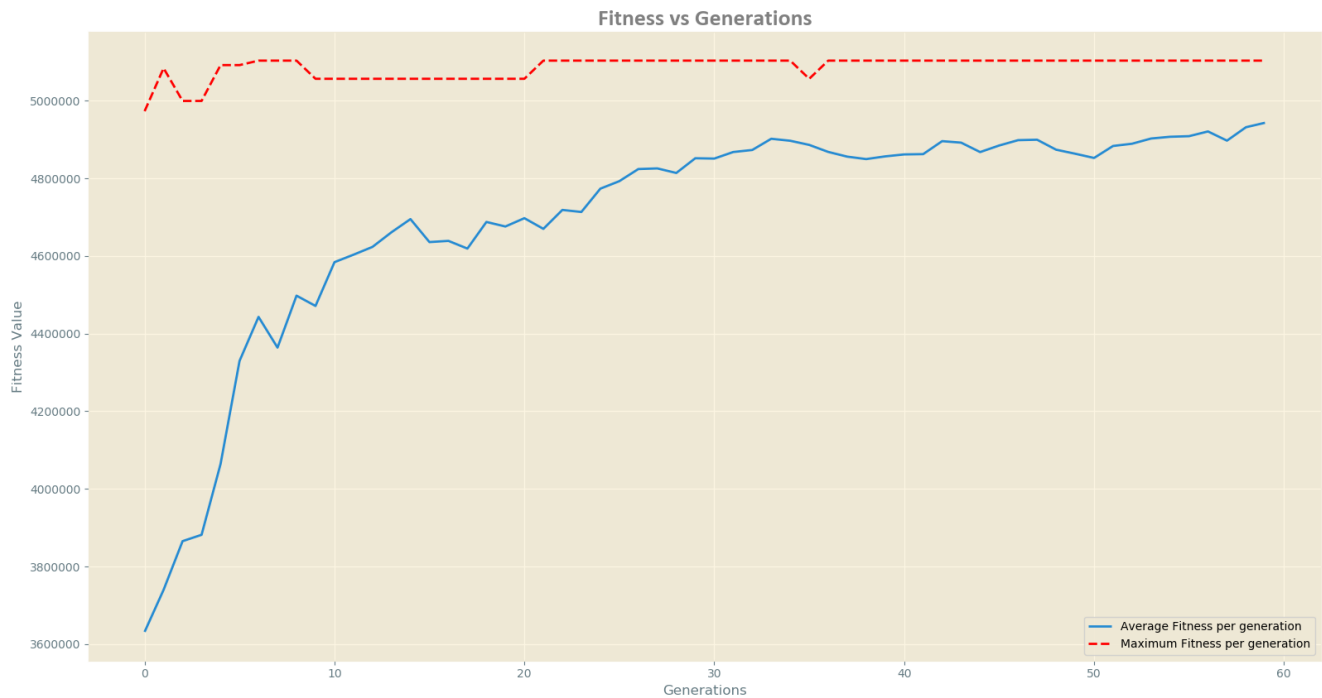


Figure 11 – Progress of the genetic algorithm. Red dotted line represents Maximum fitness per generation and blue line represents Average fitness per generation

As we are able to observe from the graph that the average fitness for a generation initially increases rapidly and with later generations, the increase slows down.

The maximum fitness value for a generation does not change much. This is due to the fact that chromosomes which have the maximum fitness have the maximum probability of advancing to the next generation and hence are maintained.

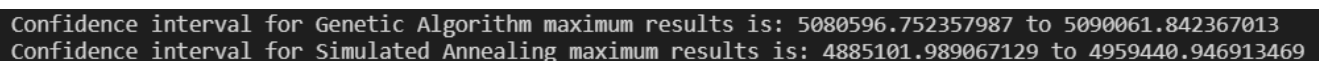
From several runs of the genetic algorithm, the optimal solution for the given data was found at [0, 0, 1, 1, 1, 0, 1, 1, 1, 0] with a fitness value of 5103165.20287.

This means that in this loan allocation solution, borrowers 3, 4, 5, 7, 8, 9 receive the loans and this maximizes our fitness function. Therefore, this is our required solution for the given data.

Comparison with another optimization model (Simulated Annealing)

On implementing the simulated annealing model and visualizing the fitness function values for each solution, it was firstly observed that the simulated annealing model generates neighbouring solutions whose fitness values vary a lot from previous. This leads to a greater number of temperature reductions being required and therefore slows down the process of finding the optimal solution. Moreover, it is observed that in simulated annealing the progress of the model is gets lost in case a much worse neighbouring solution gets accepted. This again compromises our optimal results and increases the number of temperature reductions required to reach the optimal solution therefore again increasing the time required to reach an optimal solution. This is inappropriate as though in this project, the data provided consisted of 10 borrowers meaning a solution length of 10, for banks having millions of potential borrowers, such an algorithm may slow down the process of finding the optimal lending solution. Therefore, we find that the genetic algorithm-based model for determining the optimal bank lending decision is a good way of solving the problem.

Both the algorithms were run for 100 iterations and their maximum fitness value was obtained for each iteration. On the basis of these maximum values obtained, a confidence interval was constructed for both the algorithms' results. The confidence intervals can be seen below in Figure 12:



Confidence interval for Genetic Algorithm maximum results is: 5080596.752357987 to 5090061.842367013
Confidence interval for Simulated Annealing maximum results is: 4885101.989067129 to 4959440.946913469

Figure 12 – Confidence intervals for maximum results obtained from both algorithms

As we can see, the confidence interval for the GA is smaller than the Simulated Annealing algorithm. Moreover, the confidence interval for the GA algorithm lies above the confidence interval for the Simulated Annealing algorithm meaning that it outputs results closer to the optimal solution more often. Therefore, we can conclude that GAMCC is a good method for optimizing bank lending decisions.

References

- 1) Teimouri, S., & Dutta, N. (2016). Investment and bank credit recovery after banking crises. Journal of Financial Stability, 26,306–327.
(<http://dx.doi.org/10.1016/j.jfs.2016.07.013>).
- 2) Bhargava, S. (2013). A note on evolutionary algorithms and its applications. Adults Learning Mathematics: An International Journal, 8 (1), 31–45.
- 3) Metawa, N., Hassan, M. and Elhoseny, M., 2017. Genetic Algorithm Based Model For Optimizing Bank Lending Decisions.

Annexure

In the research paper [3], the fitness function was mentioned as below in Figure 13

$$F_x = \vartheta + \varpi - \beta - \sum_{i=0}^n \lambda$$

Figure 13 – Fitness function according to the research paper [3]

However, on using this fitness function, the model was not converging towards an optimal solution. Moreover, it is mentioned in the paper that

“The GA’s fitness function (F_x) simply consists **loan revenue** (ϑ), **loans cost** (μ), **total transaction cost** (ϖ), and **cost of demand deposit** (β).”

Consequently, it is natural to assume that in Figure 13, the loans cost termed may have been missed out. On adding this term, the model converged easily towards the optimal solution. Therefore, it may be considered that the final fitness function should be

$$F_x = \vartheta + \mu + \varpi - \beta - \sum_{i=1}^n \lambda$$

And not as shown in Figure 13.

Contact Details

Name: Divayum Gupta
Roll number: 17IM30010
Department of Industrial and Systems Engineering
IIT Kharagpur
divayumgupta@gmail.com