INDIAN INSTITUTE OF TECHNOLOGY, KHARAGPUR
DEPARTMENT OF INDUSTRIAL AND SYSTEMS ENGINEERING

OPTIMISATION AND HEURISTIC METHOD LAB (IM39003)

TERM PROJECT

Optimizing Bank Lending Decisions Using Metaheuristics

Name: Priya Sinha
Roll No.: 18IM30016

Genetic Algorithm

Introduction:

Here we are using genetic algorithm for optimizing the Bank lending decisions. A Genetic Algorithm model that facilitates how banks would make an efficient decision while staying focus on the main objective of bank profit maximization.

Objective:

- To maximize the Profit.

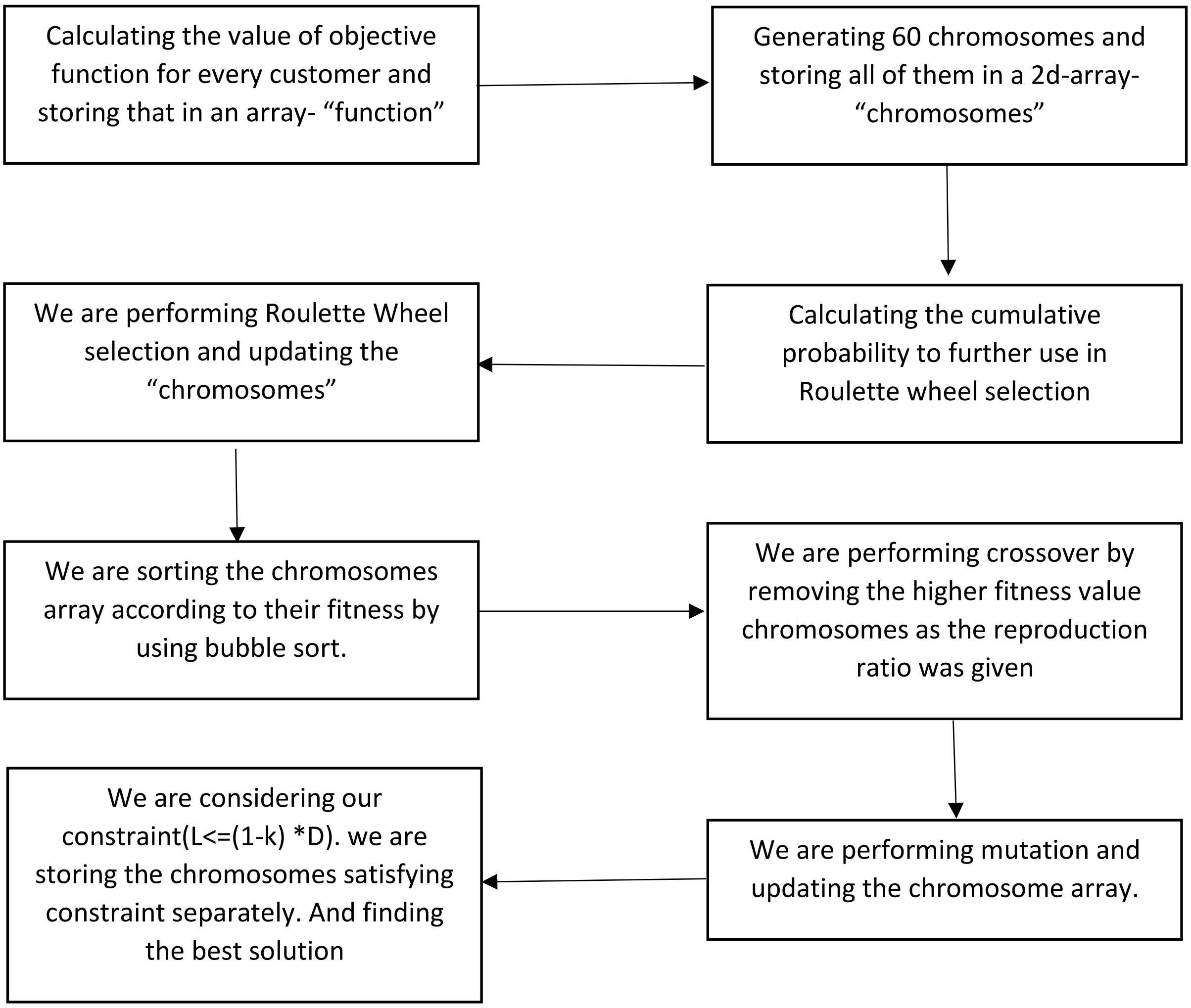
Data:

D	60									
K	0.15									
Loan Size	10	25	4	11	18	3	17	15	9	10
Interest	0.021	0.022	0.021	0.027	0.025	0.026	0.023	0.021	0.028	0.022
Rating	AAA	BB	A	AA	BBB	AAA	BB	AAA	A	A
Loss (λ)	0.0002	0.0058	0.0001	0.0003	0.0024	0.0002	0.0058	0.0002	0.001	0.001

Objective Function: $Fx = \vartheta + \varpi - \beta - \sum_{i=0}^{n}(\lambda)$
Loan Revenue: $\vartheta = \sum_{i=0}^{n}(rLL - \lambda)$
Loan Cost: $\mu = \sum_{i=0}^{n}(L\delta)$
Total transaction cost: $\varpi = \sum_{i=0}^{n}(rLT)$
Cost of demand deposit: $\beta = rDD$
Constraint: L<= (1-k) *D

Objective	- Determine a bank lending decision that maximizes the bank profit. - Determine a bank lending decision that minimizes the crediting cost.
Raw Fitness	$F_x = \vartheta + \varpi - \beta - \sum_{i=0}^{n} \lambda$
Parameters	- Population Size $(n) = 60$ - GA Generations $(\Re) = 60$ - Crossover ratio $= 0.8$ - Mutation ratio $= 0.006$ - Reproduction ratio$(n) = 0.194$ - Basic selection method is spanning the weighted roulette wheel selection
Stopping Criteria	-The evolution continues until: $F_x(t) = \Re$ where t is the time interval

Flow Chart:



Steps:

Step 1: *Imported libraries **numpy, random, math***

Step 2: *Declared values for **N, population_size, D, k, insti_cost, rD***

Step 3: *Stored the values for **loan size, loan interest rate, loss***

Step 4: *Created empty arrays to store the values of **loan_revenue, loan_cost, total_transaction_cost, cost_demand_deposit, function***

Step 5: *Got profit value as output for every customer.*

Step 6: *Generating **60 chromosomes** and storing them in 2d array*

Step 7: *Calculating the cumulative probability for each chromosome and storing them in array **cum_fit***

Step 8: *Applying **roulette wheel selection** and updating the value of chromosomes in array.*

Step 9: *Sorting array according to their fitness value (from low to high) using **bubble sort***

Step 10: *Applying crossover on the rest of chromosomes which do not undergo reproduction.*

Step 11: *After crossover applying mutation on those chromosomes*

Step 12: *Eliminating chromosomes which violates the **constraint (L <= (1-k) *D)***

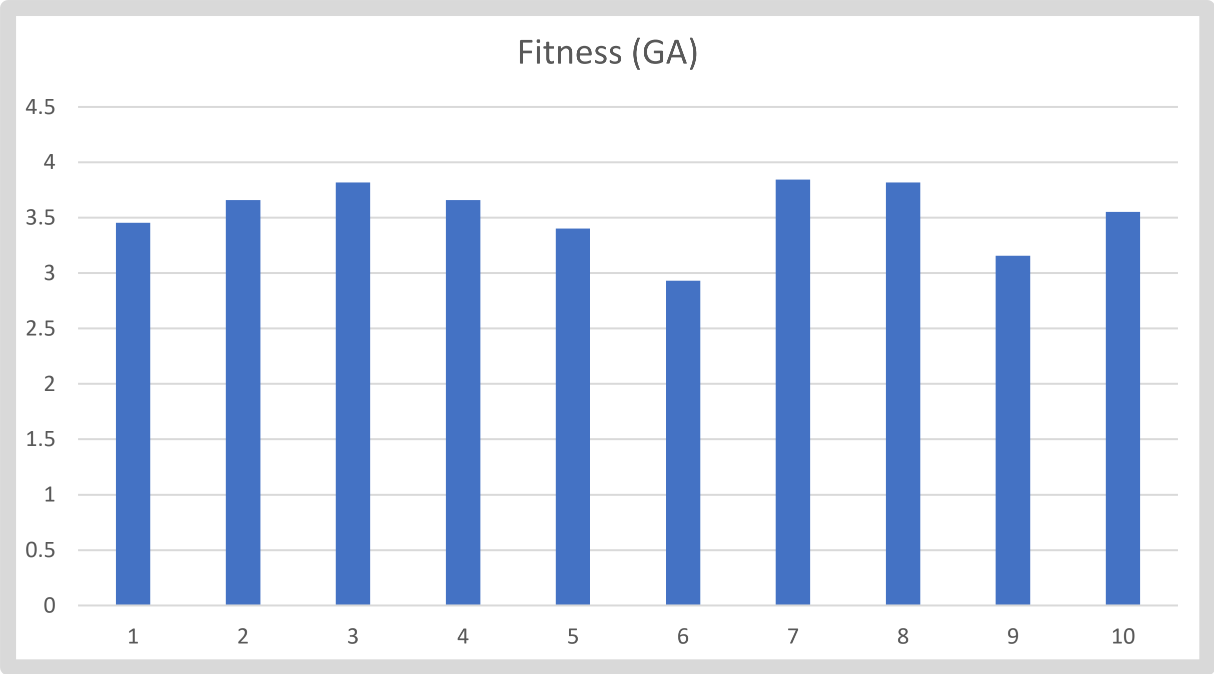
Step 13: *Storing the rest of chromosomes in **chromo** and fitness value in **fit.***

Step 14: *Getting the maximum fitness value from **fit** and corresponding chromosome from **chromo***

Result and Inference:

Running the code for 10 times and noting the best solution and its fitness.

Chromosomes	Fitness	Loan Size
[0 0 1 1 0 0 1 0 1 1]	3.4546	51
[0 0 1 1 0 1 1 0 1 0]	3.6602	44
[0 0 0 1 1 1 0 0 1 1]	3.8182	51
[1 0 0 1 0 1 1 0 1 0]	3.66	50
[0 0 1 0 0 1 1 0 1 1]	3.4038	43
[0 0 0 0 1 1 0 1 1 0]	2.9324	45
[1 0 1 0 0 1 0 1 1 1]	3.8436	51
[0 0 0 1 1 1 0 0 1 1]	3.8182	51
[0 0 1 0 1 1 0 1 0 1]	3.1572	50
[0 0 1 0 1 1 1 0 1 0]	3.554	51



Best solution obtained so far:

Chromosomes: [1 0 1 0 0 1 0 1 1 1]
Fitness: 3.8436

<u>Simulated Annealing</u>

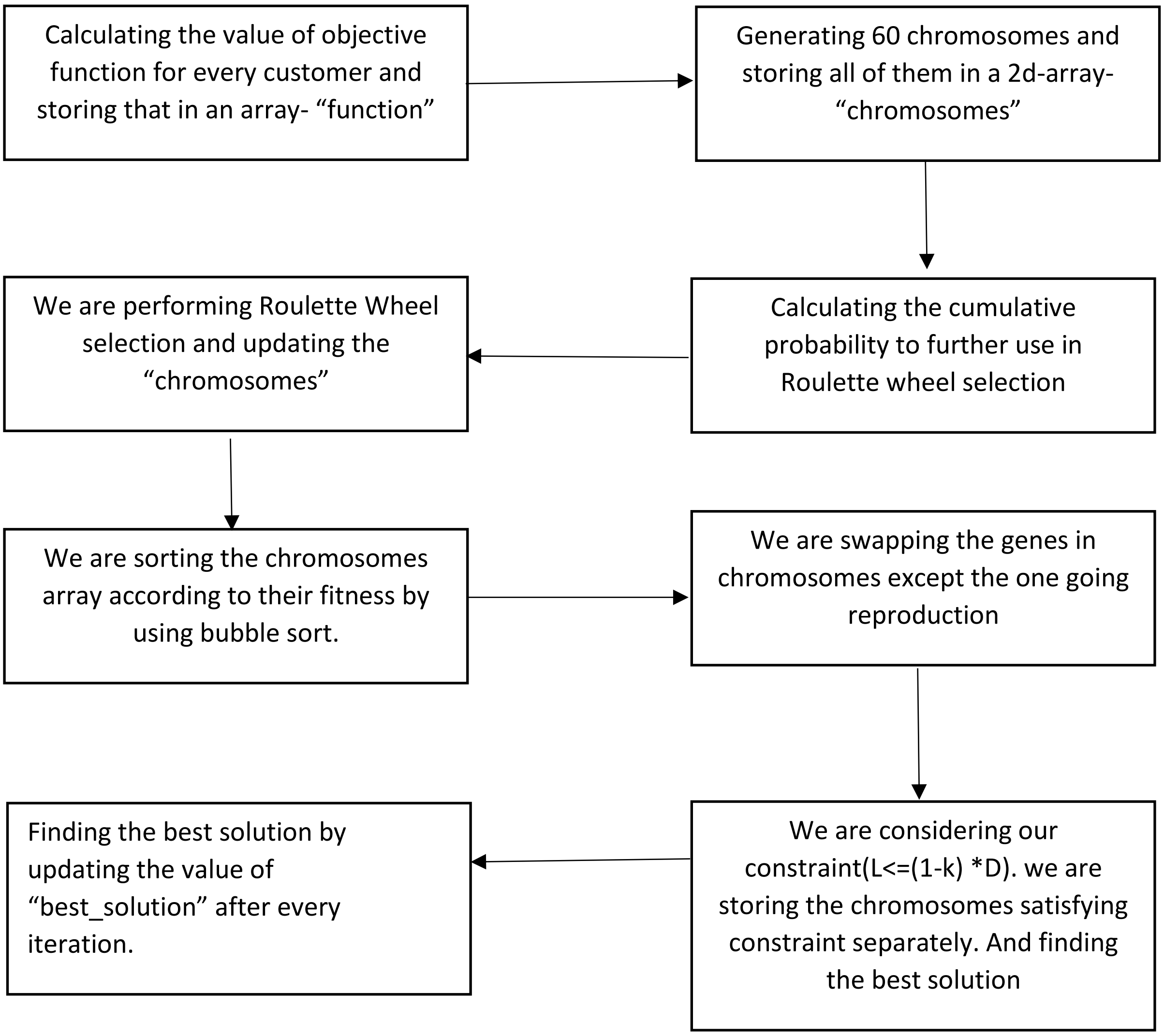
Introduction:
Here we are using simulated annealing for optimizing the Bank lending decisions. A SA model that facilitates how banks would make an efficient decision while staying focus on the main objective of bank profit maximization.

Objective:

- To maximize the Profit.

Parameters:
Initial Temperature: T0 = 300
Decrease in temperature after every iteration = -5

Flow Chart:



Steps:

Step 1: *Imported libraries* ***numpy, random, math***

Step 2: *Declared values for* ***N, population_size, D, k, insti_cost, rD***

Step 3: *Stored the values for* ***loan size, loan interest rate, loss***

Step 4: *Created empty arrays to store the values of* ***loan_revenue, loan_cost, total_transaction_cost, cost_demand_deposit, function***

Step 5: *Got profit value as output for every customer.*

Step 6: *Generating* ***60 chromosomes*** *and storing them in 2d array*

Step 7: *Calculating the cumulative probability for each chromosome and storing them in array* ***cum_fit***

Step 8: *Applying* ***roulette wheel selection*** *and updating the value of chromosomes in array.*

Step 9: *Sorting array according to their fitness value (from low to high) using* ***bubble sort***

Step 10: *Declaring* ***T0, Tf, n***

Step 11: *Swapping the randomly chosen bits and updating array* ***chromosome***

Step 12: *Eliminating chromosomes which violates the* ***constraint (L <= (1-k) *D)***

Step 13: *Storing the rest of chromosomes in* ***chromo*** *and fitness value in* ***fitness.***

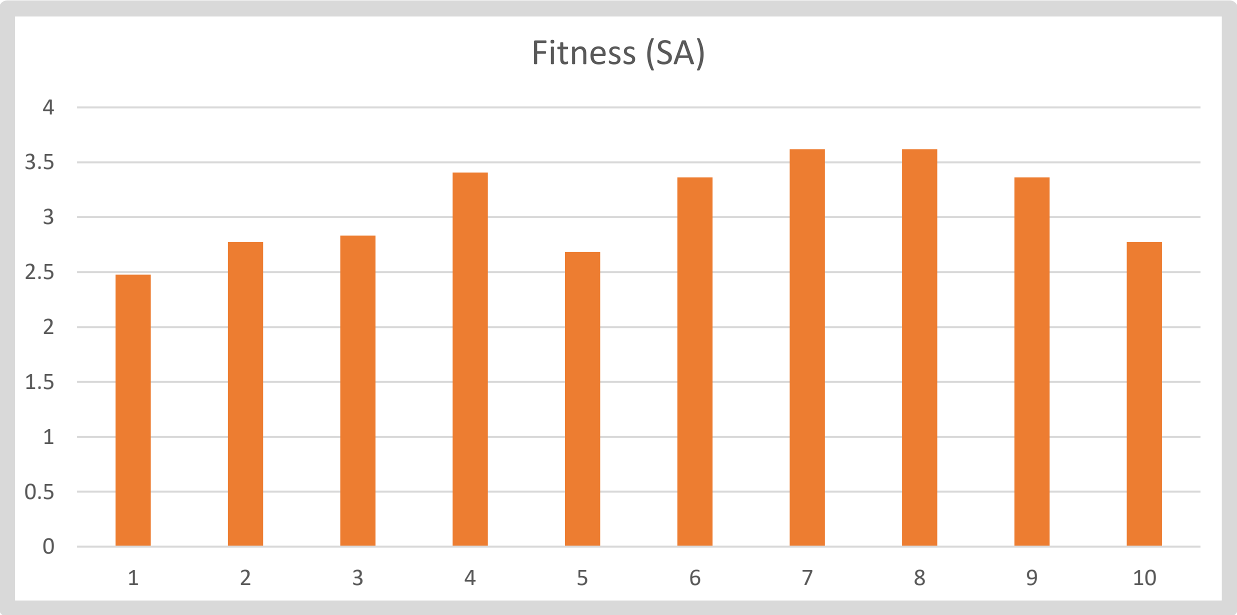
Step 14: *declaring* ***best_solution*** *and applying simulated annealing and updating the value of best solution accordingly.*

Step 15: *Printing the value of best solution and respective chromosome.*

Result and Inference:

Running the code for 10 times and noting the best solution and its fitness.

Chromosomes	Fitness	Loan Size
[0 0 1 1 0 0 0 1 0 1]	2.4778	40
[0 0 0 1 0 1 1 1 0 0]	2.774	46
[0 0 1 1 0 0 0 0 1 1]	2.8332	34
[1 0 1 1 0 0 1 0 1 0]	3.4052	51
[1 0 0 1 0 1 0 1 0 0]	2.6832	39
[0 0 1 1 0 0 0 1 1 1]	3.3638	51
[0 0 0 1 0 1 0 1 1 1]	3.6186	48
[0 0 0 1 0 1 0 1 1 1]	3.6186	48
[0 0 1 1 0 0 0 1 1 1]	3.3638	49
[0 0 0 1 0 1 1 1 0 0]	2.774	46

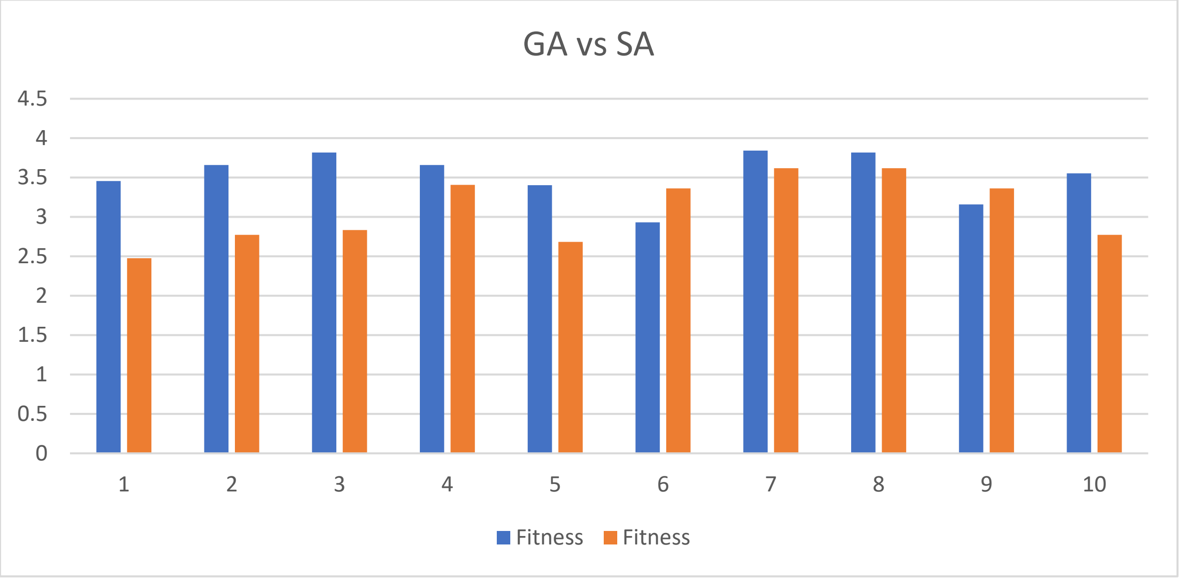


Best solution obtained so far:

Chromosomes: [0 0 0 1 0 1 0 1 1 1]
Fitness: 3.6186

Comparison:

Orange- Fitness SA
Blue- Fitness GA



From here we can clearly see that ***Genetic Algorithm*** is giving better result than ***Simulated Annealing.***

Genetic Algorithm & Simulated Annealing

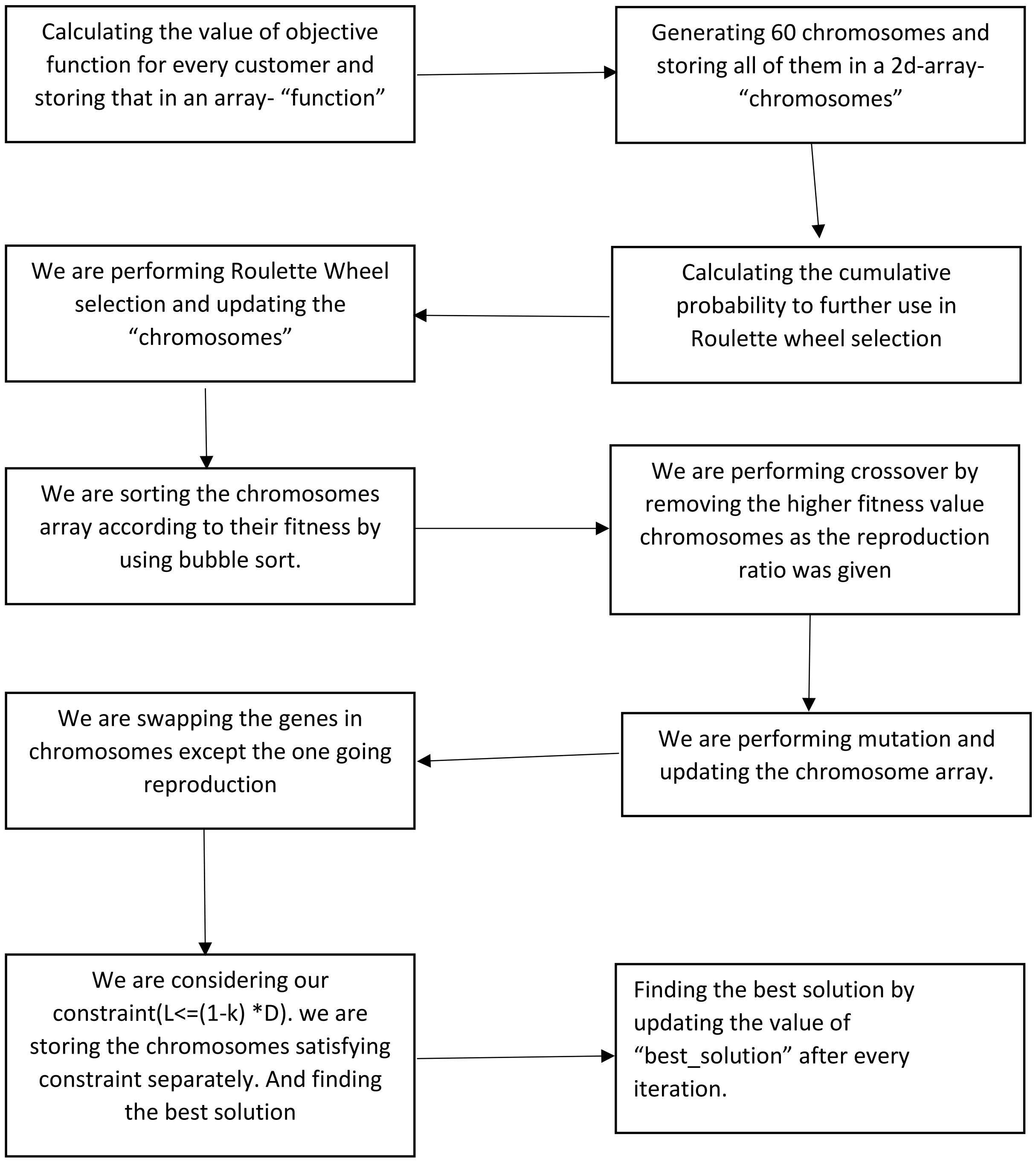
Introduction:
Here we are using genetic algorithm and simulated annealing for optimizing the Bank lending decisions. A GA & SA model that facilitates how banks would make an efficient decision while staying focus on the main objective of bank profit maximization.

Objective:

- To maximize the Profit.

Parameters:
Initial Temperature: T0 = 300
Decrease in temperature after every iteration = -5
Crossover Probability = 0.8
Mutation Probability = 0.006
Reproduction Ratio = 0.194

Flow Chart:


Calculating the value of objective function for every customer and storing that in an array- “function”
Generating 60 chromosomes and storing all of them in a 2d-array- “chromosomes”
Calculating the cumulative probability to further use in Roulette wheel selection
We are performing Roulette Wheel selection and updating the “chromosomes”
We are sorting the chromosomes array according to their fitness by using bubble sort.
We are performing crossover by removing the higher fitness value chromosomes as the reproduction ratio was given
We are performing mutation and updating the chromosome array.
We are swapping the genes in chromosomes except the one going reproduction
We are considering our constraint(L<=(1-k) *D). we are storing the chromosomes satisfying constraint separately. And finding the best solution
Finding the best solution by updating the value of “best_solution” after every iteration.

Steps:

Step 1: *Imported libraries* ***numpy, random, math***

Step 2: *Declared values for* ***N, population_size, D, k, insti_cost, rD***

Step 3: *Stored the values for* ***loan size, loan interest rate, loss***

Step 4: *Created empty arrays to store the values of* ***loan_revenue, loan_cost, total_transaction_cost, cost_demand_deposit, function***

Step 5: *Got profit value as output for every customer.*

Step 6: *Generating* ***60 chromosomes*** *and storing them in 2d array*

Step 7: *Calculating the cumulative probability for each chromosome and storing them in array* ***cum_fit***

Step 8: *Applying* ***roulette wheel selection*** *and updating the value of chromosomes in array.*

Step 9: *Sorting array according to their fitness value (from low to high) using* ***bubble sort***

Step 10: *Applying crossover on the rest of chromosomes which do not undergo reproduction.*

Step 11: *After crossover applying mutation on those chromosomes*

Step 12: *Declaring* ***T0, Tf, n***

Step 13: *Swapping the randomly chosen bits and updating array* ***chromosome***

Step 14: *Eliminating chromosomes which violates the* ***constraint (L <= (1-k) *D)***

Step 15: *Storing the rest of chromosomes in* ***chromo*** *and fitness value in* ***fitness.***

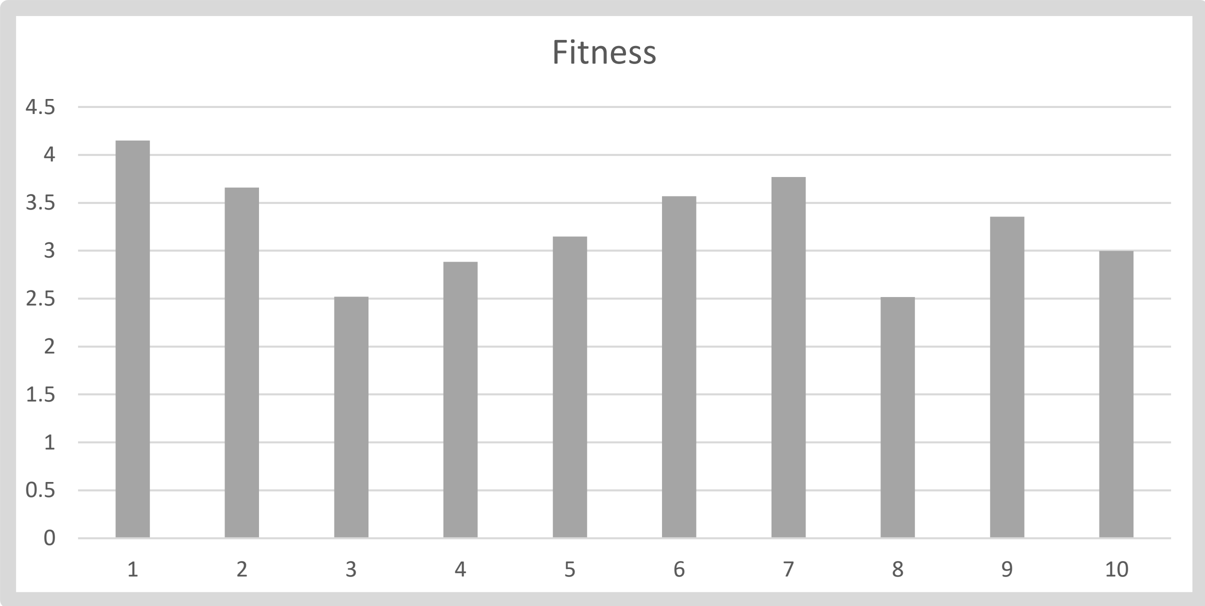
Step 16: *declaring* ***best_solution*** *and applying simulated annealing and updating the value of best solution accordingly.*

Step 17: *Printing the value of best solution and respective chromosome.*

Result and Inference:

Running the code for 10 times and noting the best solution and its fitness.

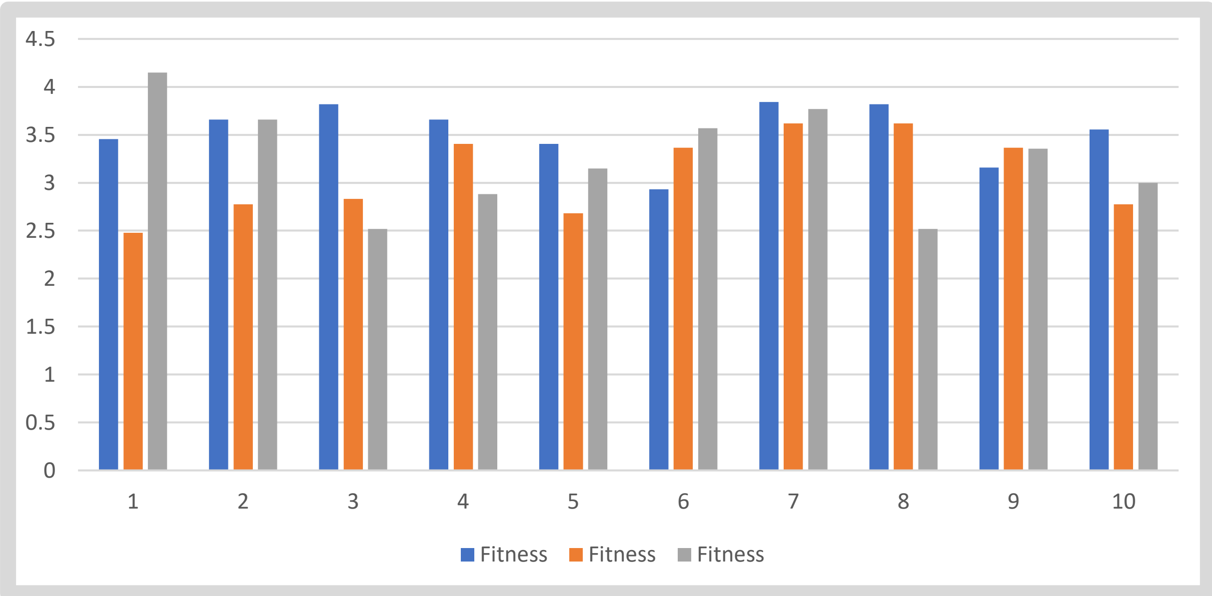
Chromosomes	Fitness	Loan Size
[1 0 1 1 0 1 0 0 1 1]	4.1494	47
[1 0 0 1 0 1 1 0 1 0]	3.66	50
[0 0 1 1 0 0 1 1 0 0]	2.5192	47
[1 0 0 1 1 1 0 0 0 0]	2.8828	42
[1 0 1 0 0 0 1 0 1 1]	3.1488	50
[0 0 1 1 0 1 0 1 1 0]	3.5694	42
[0 0 1 1 1 1 0 0 1 0]	3.769	45
[0 1 1 1 0 0 0 0 0 1]	2.5176	50
[0 0 1 0 0 1 1 1 1 0]	3.3544	48
[1 0 1 0 0 1 1 1 0 0]	2.999	49



Best solution obtained so far:
Chromosomes: [1 0 1 1 0 1 0 0 1 1]
Fitness: 4.1494

Comparison:

Orange- Fitness SA
Blue- Fitness GA
Grey- Fitness GA & SA



Average fitness GA = 3.53022
Average fitness SA = 3.09122
Average fitness GA & SA = 3.25696
From here we can say using GA & SA gives better than SA but not than GA.
GA >= GA & SA >= SA