Shashank Sundi
19IM10028

Department of Industrial and Systems Engineering
Indian Institute of Technology, Kharagpur
OHM Term Project (IM39003)

Optimizing Bank Lending Decisions Using Metaheuristics

Problem Statement

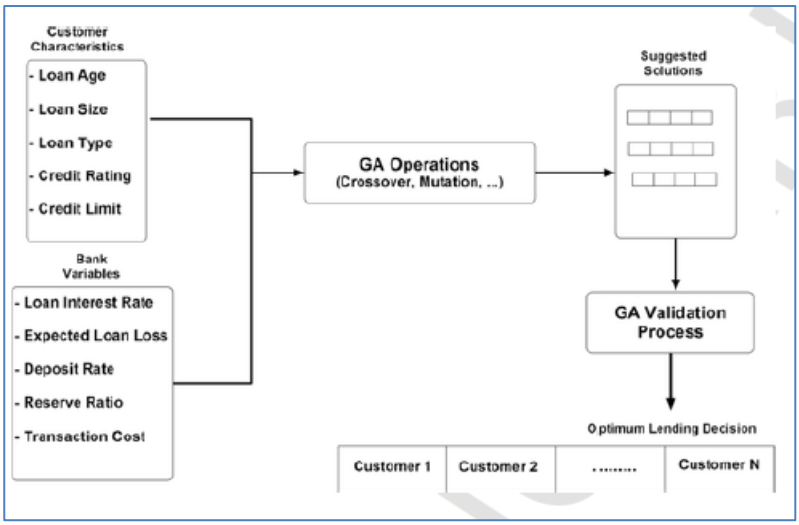
During any financial crisis, there is very limited credit available within the banking sector. This has to be managed to distribute the limited credit available in a way that maximizes their profits in the time of crisis .Therefore, there is a need to set an optimal mechanism of bank lending decisions that will maximize the bank profit in a timely manner. The paper therefore proposes a model based on Genetic Algorithm (GA) to organize bank lending decision in a highly competing environment with credit crunch constraint.

Introduction

This project deals with the implementation of Genetic Algorithms and Amalgamation of Genetic Algorithm with Simulated Annealing, to make Bank Lending Decision.
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

Methodology

In the proposed method, the lending decision is dynamically decided based on customer's loan characteristics. It is assumed that all customers are applicable (good) to get the required loan.GA is employed to search for the most suitable customers depending on a set of factors such as loan age, loan size, loan interest rate, loan type, and borrower credit rating. Moreover, bank variables that are considered for selecting this optimal solution are loan interest rate, expected loan loss ,deposit rate ,reserve ratio, transaction cost .



Variables used

1) Loan Age (α) – No. of years for repaying the loan
2) Credit Limit – Maximum loan amount a borrower is eligible for.
3) **Loan Size (L)** – amount of loan requested by a specific customer
4) 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.
5) Expected Loan Loss (λ) – Borrow Credit Rating is used to measure the range of the expected

Loan size categories.

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

The credit rating and the corresponding expected loan loss.

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

Loan interest rate assignment.

φ	α	r_L Value
M	1	NA
	2	NA
	3	NA
	4	$0.021 \le r_L \le 0.028$
P	1	$0.0599 \le r_L \le 0.0601$
	2	$0.0601 < r_L \le 0.0604$
	3	$0.0604 < r_L \le 0.0609$
A	1	$0.0339 \le r_L \le 0.03349$
	2	$0.0349 < r_L \le 0.0379$
	3	$0.0379 < r_L \le 0.0399$

Data to be used

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

The above data is to be used while implementing the GA or GA-SA algorithms.

Genetic Algorithm Fitness Function

Fitness function Fx is given by the formula:

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

• Loan Revenue (ϑ)
The value of the loan revenue is calculated using the loan interest rate (rL), loan size (L), and the expected loan loss (λ).

$$\vartheta = \sum_{i=0}^{n} \left(r_L L - \lambda \right)$$

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

$$\mu = \sum_{i=0}^{n} L\delta$$

• Total Transaction Cost (ϖ)
The value of the total transaction cost is determined using institute transactional cost (T) and the customer transaction rate (rT).
The value of rT has been assumed to be 0.01 for the purpose of this project.
Here, the institute transactional cost (T) is determined using the below formula:

T = (1-K)D

$$\varpi = \sum_{i=0}^{n} r_L T$$

• Cost of Demand deposit (β)

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

$$\beta = r_D D$$

GA Validation Process to Generate a GA population with n chromosomes

In the given data, we have 10 customers, so size of each chromosome is 1x10 , where each bit consists of 1 or 0 depending upon, whether the customer has been lent the loan or not. When we initially generate a population of chromosomes , many of them might be exceeding the credit constraints of the banks .The chromosomes which do not satisfy the credit crunch constraint are discarder . 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.

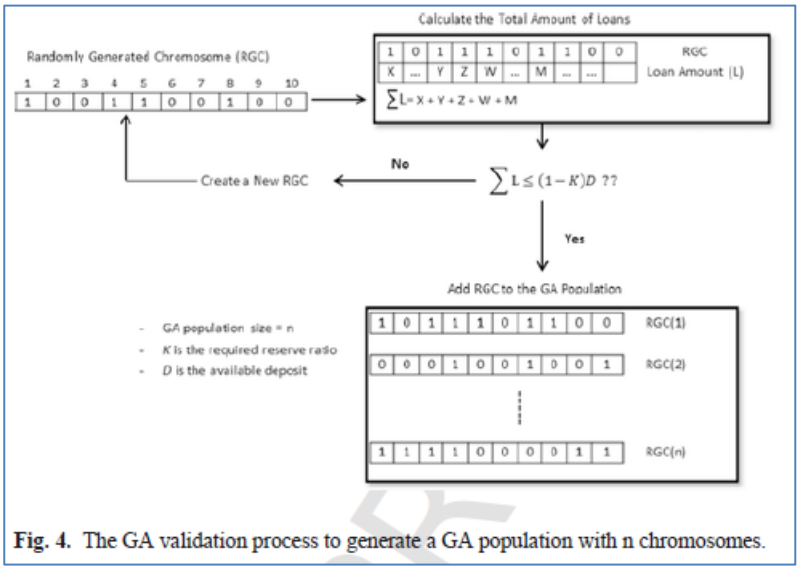


Fig. 4. The GA validation process to generate a GA population with n chromosomes.

Genetic Algorithm Pseudo Code

- Generate Initial Population that satisfies GA Validation Process.
- For n iterations:
  - -Evaluate Fitness of Each Lending Decision in the population
  - -Perform Roulette Wheel Selection to get Parent Pool
  - -Create New Generation Population using Reproduction, Crossover and Mutation
  - -Perform Forceful Crossover in case no solution generated in new generation
  - -Check Validity of solutions in New Generation Population
  - -Store Best Solution of each generation
- Plot and display best Solution

GA Results

"Best Fitness Value " "6.8352"

"Best Bank Lending…" "1" "0" "1" "1" "0" "1" "0" "0" "1" "1"

So, we should lend money to

" Customer" "1" "--Rating" "AAA"

" Customer" "3" "--Rating" "A"

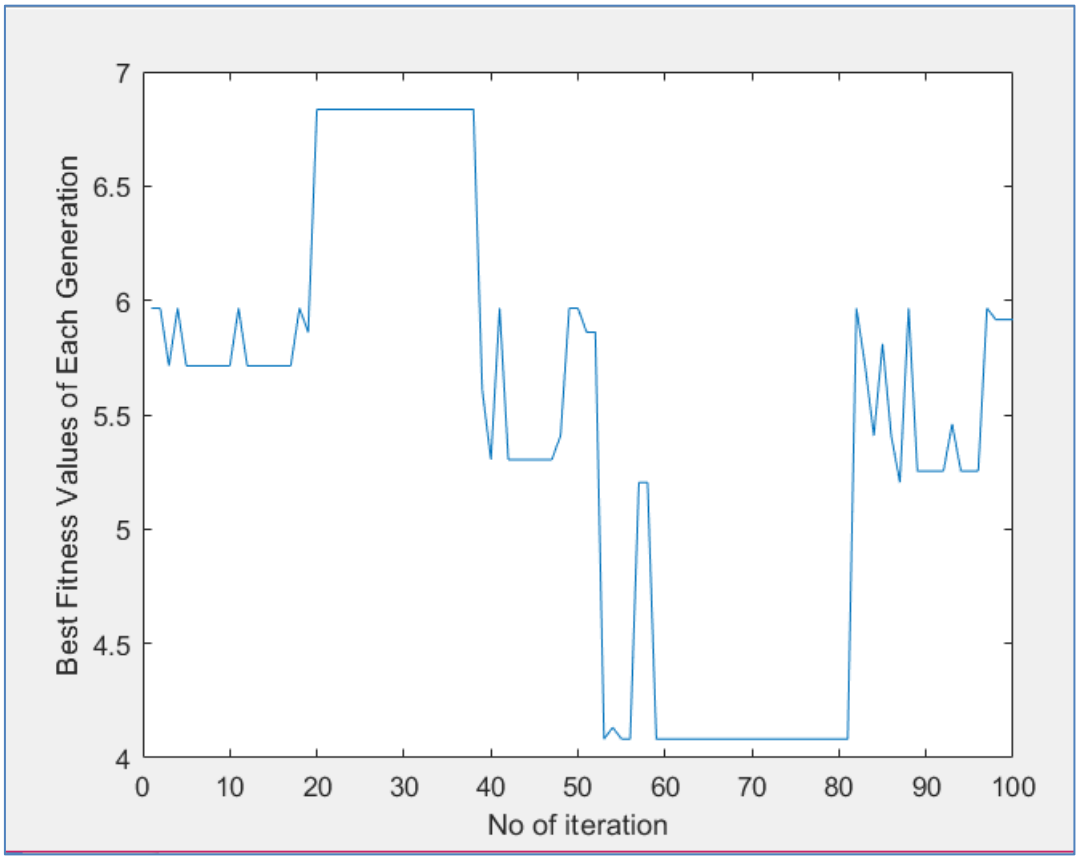
" Customer" "4" "--Rating" "AA"

" Customer" "6" "--Rating" "AAA"

" Customer" "9" "--Rating" "A"

" Customer" "10" "--Rating" "A"

```
Command Window
    "Best Fitness Value "    "6.8352"

    "Best Bank Lending..."    "1"    "0"    "1"    "1"    "0"    "1"    "0"    "0"    "1"    "1"

So, we should lend money to
    " Customer"    "1"    "--Rating"    "AAA"

    " Customer"    "3"    "--Rating"    "A"

    " Customer"    "4"    "--Rating"    "AA"

    " Customer"    "6"    "--Rating"    "AAA"

    " Customer"    "9"    "--Rating"    "A"

    " Customer"    "10"    "--Rating"    "A"
```



Genetic Algorithm-Simulated Annealing(GA-SA) Pseudo Code

- Generate Initial Population that satisfies GA Validation Process.
- For n iterations:
  - -Evaluate Fitness of Each Lending Decision in the population
  - -Perform Roulette Wheel Selection to get Parent Pool
  - -Create New Generation Population using Reproduction, Crossover and Mutation
  - -Perform Forceful Crossover in case no solution generated in new generation
  - -Simulated Annealing Loop
  - While T>Tmin:
    - For k iterations:
      - For each chromosome:
        - -generate neighbourhood solution by swapping bits
        - -calculate fitness and compare with fitness of parent
        - -if fitness of neighbour < fitness of parent:
          - Accept solution with Metropolis Criterion
        - -else keep original sol.
    - -T=t*c --- cooling
  - -Check Validity of solutions in New Generation Population
  - -Store Best Solution of each generation
- Plot and display best Solution

GA-SA Results

"Best Fitness Value " "6.5293"

"Best Bank Lending…" "1" "0" "1" "0" "0" "1" "0" "1" "1" "1"

So, we should lend money to

" Customer" "1" "Rating" "AAA"

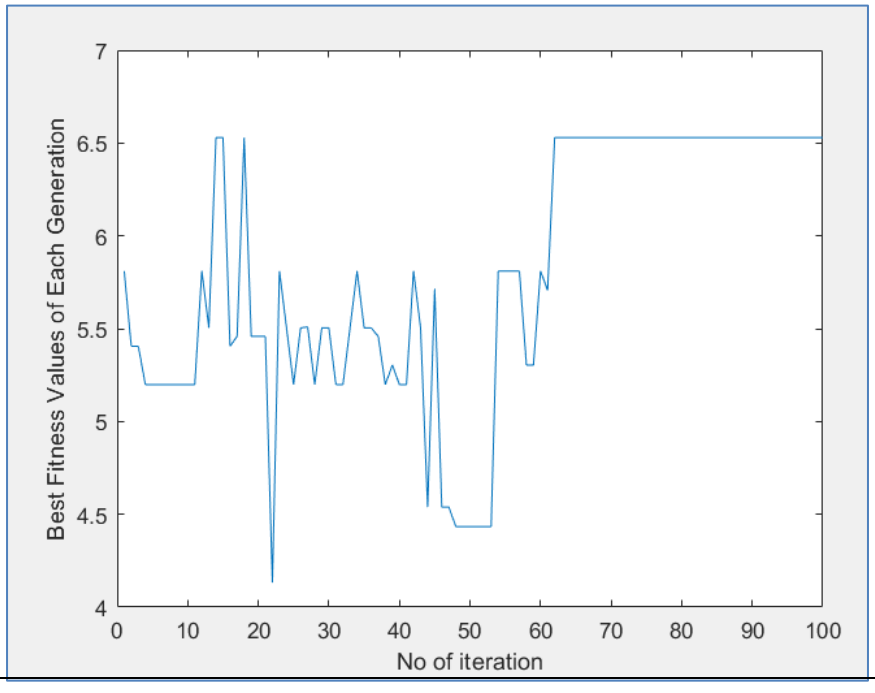
" Customer" "3" "Rating" "A"

" Customer" "6" "Rating" "AAA"

" Customer" "8" "Rating" "AAA"

" Customer" "9" "Rating" "A"

" Customer" "10" "Rating" "A"



Comparison of GA and GA-SA-amalgamation

- It is observed that GA gives better solution than GA-SA. The best fitness value obtained using GA is 6.8352 whereas best fitness using GA-SA is 6.5293.
- Using GA , the loan is lent to customers – 1,3,4,6,9,10 and using GA-SA we get customers 1,3,6,8,9,10.
- The solution in GA converges faster , but it starts to fluctuate wildly after some iterations. However, in GA-SA the solution initially fluctuates and then reaches convergence with increase in iterations.
- Thus we can conclude that although GA gives better solutions , it is not as stable as GA-SA. Both algorithms give good results and are hence good algorithms to be used , depending on the use case.