GENETIC ALGORITHM FOR OPTIMAL BANK LENDING DECISIONS

OPTIMISATION AND HEURISTIC METHODS PROJECT (IM39003)

Mintu Agarwal
18IM10034
Industrial and Systems Engineering Department
IIT Kharagpur

CONTENTS

PROBLEM STATEMENT

At the time of financial crisis, a bank's ability to continue with its traditional bank lending strategies are fore concerned. During a financial crisis, the most important problem is how will a bank manage to distribute the limited credit available in a way that maximizes their profits in the time of crisis.

The insolvency of some banks can lead to a reduction in bank loans, not only because of the bad bank failures but also because the healthy banks tend to be more cautious in their lending practices through cutting back on lending.

The inability of banks to manage loan portfolios efficiently may result in a credit crunch. A credit crunch is often caused by a sustained period of careless and inappropriate lending, resulting in losses for lending institutions and investors in debt when the loans turn sour and the full extent of bad debts becomes known. These challenges have led to a rise in more formal and accurate methods and models to optimize the lending decision and minimize loan risks.

The goal is to make lending decisions in a credit crunch environment where all applicable borrowers are eligible to get the desired loans. This is an NP-hard optimization problem which can be solved using meta-heuristic algorithms such as population-based algorithms.

The Paper proposes an intelligent model based on Genetic Algorithm (GA) to organize bank lending decision.

Why Genetic Algorithms?

Multiple factors including loan characteristics, creditor ratings and expected loan loss can be easily integrated to GA chromosomes. Each borrower can, therefore, be represented by a gene in the GA chromosome.

The Genetic Algorithm (GA) model aims to systematically stabilise banks while achieving maximum profit and to establish the capital base so that banks can increase lending efficiently.

Problem Representation

The basic assumption for the model is that the 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:

- Loan Age (α): With an assumption that the maximum number of years for any loan is 20 years, the loan age is divided into four categories depending on its expected period time.

Loan age categories.

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

- Loan Size (L): Depending on the credit limit, the loan size is determined. The loan size determines the amount of loan requested by a specific customer.

Loan size categories.

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	$\$ 100{,}001 < \alpha \leq \$ 250{,}000$

- Loan Type (φ): There are three types of loans assumed: Mortgage (M), Personal (P), and Auto (A).

- Credit Rating: Borrow Credit Rating is used to measure the range of the expected loan loss.

The credit rating and the corresponding expected 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	$0.0058 < \lambda \leq 0.0119$

- Credit Limit: The credit limit represents the maximum loan amount that can be given to the customer based on his income and occupation.

- Loan interest rate (r_L): Based on the values of φ and α, the loan interest rate r_L is assigned.

- Expected Loan Loss (λ): The expected loan loss from different borrowers is calculated by the credit rating assigned to them.

Encoding

The genes of the chromosomes are binary encoded with the length of string being the number of eligible borrowers. The value stored is each gene is Boolean. If the loan is provided to a particular borrower then the value of the gene is 0 for that solution, 1 otherwise. 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.

The figure below is an example of such chromosomes.

Fitness Function

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

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

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

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

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

3) Total Transaction Cost (omega): 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.

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

4) Cost of Demand deposit (β): The value is determined using the bank's deposit interest rate (r_D) and the bank's deposit (D).

$$\beta = r_D D$$

Final Fitness function (Fx):

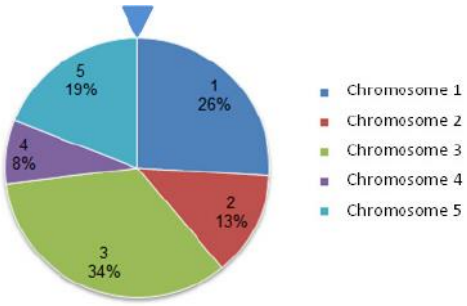
$$F_x = \vartheta + \varpi - \beta - \sum_{i=0}^{n} \lambda$$

The Genetic Algorithm

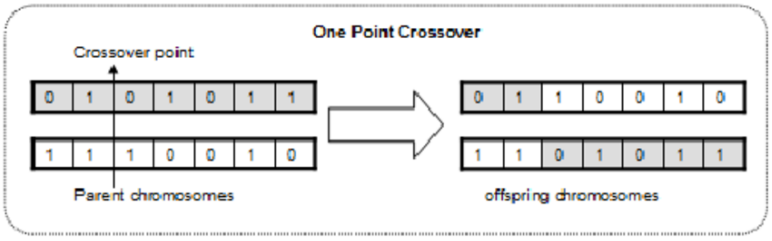
GA is a evolutionary algorithm which simulates some of the natural processes: selection, inheritance via genetic crossover and random mutation.

After generation of initial solutions, the next steps in GA are as follows:

1) Selection: Selection of parent pool for reproduction based on the fitness values of the current generation individual. There are different ways of forming the reproduction pool like proportionate selection, roulette wheel selection, tournament selection etc. In our model, we have use roulette wheel selection.



2) Crossover: It is the process of concatenating two chromosomes to generate a new two chromosomes by switching genes. The input of this process is two chromosomes while its output is two different chromosomes which inherit the genes of the parent chromosomes. In our model, a simple single-point crossover is used by randomly obtaining a single pivot point.



3) Mutation: It is the process of randomly revers the value of one gene in a chromosome. So, the input is a single chromosome and the output is different one. This has a lot of significance on the exploration factor of the algorithm. In our model, the whole string is traversed and if the random number generated is less than the mutation probability, the Boolean value at the gene is reversed. The mutation probability is usually very low.

   Above processes are repeated until the stopping criteria is reached, which in our case is to stop after a certain number of generations.

The pseudocode adopted in the paper is as follows:

Randommly generate a pool of S Lending Decisions $X = \{x_1, x_2, \ldots, x_S\}$.
Represent each decision in S by one chromosome to get a pool P of chromosomes $U = \{u_1, u_2, \ldots, u_P\}$.
$\forall u_i \in U$, Forming the Lending Decision According to Parameters)
Evaluate the fitness of each $u_i \in U$ using F_x.
for $q = 1, 2, \ldots, Q$ **do**
  $\tilde{U} \Leftarrow \emptyset$
  for $p = 1, 2, \ldots, P$ **do**
    Randomly select $u_a, u_b \in U$ $(a \neq b)$ based on the normalized fitness $\tilde{f}(u)$:

$$\tilde{f}(u) = \frac{f(u)}{\sum_c f(u)}.$$

    Cross over u_a and u_b according to α

$$\mathcal{C}(u_a, u_b|\alpha) \Rightarrow u'_a, u'_b.$$

    Perform mutation on u'_a and u'_b according to $\vec{m}$

$$\mathcal{M}(u'_a|\beta) \Rightarrow \tilde{u}_a, \quad \mathcal{M}(u'_b|\beta) \Rightarrow \tilde{u}_b.$$

    Evaluate $f(\tilde{u}_a)$ and $f(\tilde{u}_b)$.
    $\tilde{U} \Leftarrow \tilde{U} \cup \{\tilde{u}_a, \tilde{u}_b\}$
  end for
  $U \Leftarrow \{u_i; u_i \in \tilde{U} and f(u_i)\}$
end for
Find the chromosome u^* that satisfies

$$u^* = \arg\max_u f(u), u \in U$$

Return the lending decision x^* by mapping u^* back

Code : GA_paper.py

Next, we incorporate some features of a random-search method algorithm called Simulated Annealing into the above.

AMALGAMATION OF GENETIC ALGORITHM WITH SIMULATED ANNEALING

Simulated Annealing (SA) mimics the Physical Annealing process but is used for optimizing parameters in a model. In the algorithm, we try to generate several neighbourhood solutions and accept it on following basis:

1) If the objective function value improves for the new solution, we simply accept it and replace the old solution with this value.
2) Else, we find a value called probability of acceptance and generate a random number and, if the random number generated is less than the probability of acceptance, we accept the new solution, else discard it.

The second step of the above process is important because it provides a way of exploring the solution space more whereas the first step is more of exploitation.

$$P = \begin{cases} 1 & if \Delta c \leq 0 \\ e^{-\Delta c / t} & if \Delta c > 0 \end{cases}$$

We incorporate this in our model after the crossover step of a generation. Instead of having crossover probability, we compare the maximum fitness among the off-springs to the maximum fitness of the parent and decide whether to accept the new chromosomes for the next generation or to go ahead with the current chromosomes. Also, a temperature variable is maintained which affects the acceptance probability value, at initial(high) temperature the value of P_a is high and hence, the solutions are generally accepted providing more exploration, whereas, when the temperature decreases, the first step is mostly accepted hence there is more exploitation. Therefore, the model gradually shifts from a random-search to a hill-climbing approach. There are different ways of finding the initial temperature. In our model, we find the mean of fitness and use geometric cooling as

$$T_{t+1} = \alpha T_t.$$

Code: GA_amalgamated.py

RESULTS AND COMPARISON

Comparison of some plots obtained on applying both the algorithms:

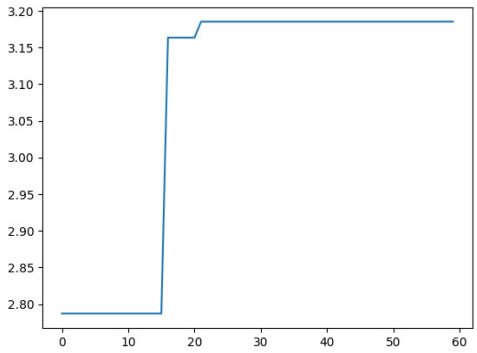


Fig. 1

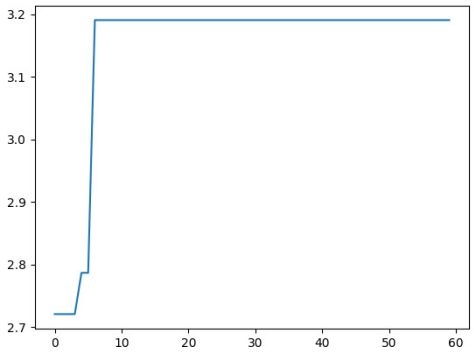


Fig. 2

On running both the algorithms, several number of times, the following observations were made:

1) Both the algorithms result in providing the optimal bank lending decision almost every time, given the number of generations is high.
2) The rate of convergence for the amalgamation was slightly higher than the original implementation of GA.
3) For more complex decisions (more genes in chromosomes), it can be argued that the second algorithm may converge faster with some parameter tuning to temperature and other factors.
4) Both the algorithms are capable of the solving the given, NP-hard optimization problem, signifying the use of evolutionary algorithms in the field of decision making.