IM39003 - Term Project

NAME - SUMIT KUMAR
ROLL - 18IM10029

PROJECT 1: Optimizing Bank Lending Decisions Using Metaheuristics

Given Data Set

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

Approach

In the first part of the solution I used a simple Genetic Algorithm to find the solution and in the second part I did amalgamation of Genetic Algorithm with Simulated Annealing to reach the best solution.

GA Fitness Function

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

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

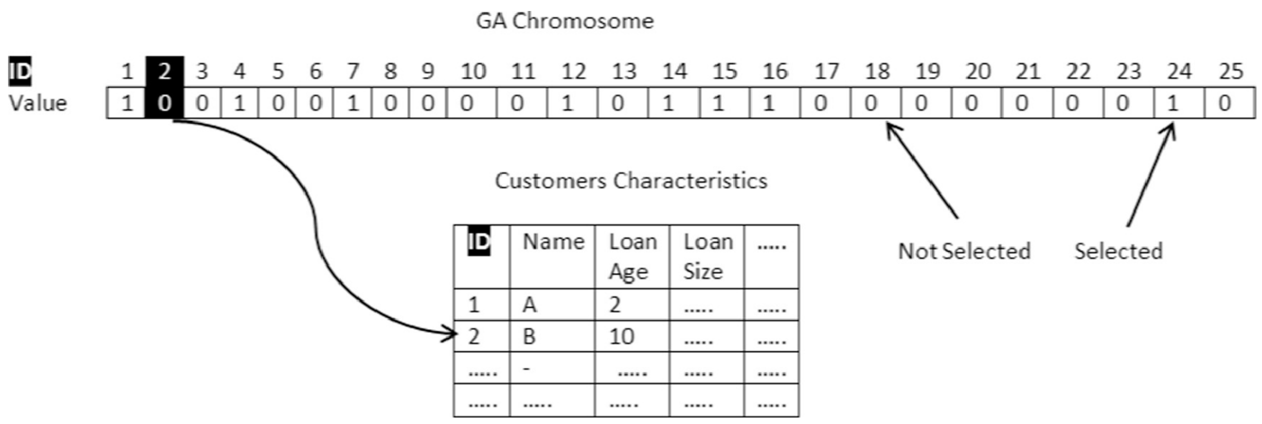
$$\varpi = \sum_{i=0}^{n} r_L T$$
$$T = (1 - K)D - L$$
$$\beta = r_D D$$

Constraint for Chromosomes

$$\sum \mathrm{L} \leq (1 - K)D$$

Where,

Fx = Fitness Function
rL= Interest Rate
L= Loan Size
λ = Expected Loan Loss
K = Reserved Ratio
D = Deposit
rT = Customer Transaction Rate (0.01)
rD = Weighted Average of Deposit Rates(0.9%)



ID	Name	Loan Age	Loan Size	.....
1	A	2	.....	.....
2	B	10	.....	.....
.....	-	.....	.....	.....
.....	.....	.....	.....	.....

Fig. 1. Chromosome representation for 25 customers.

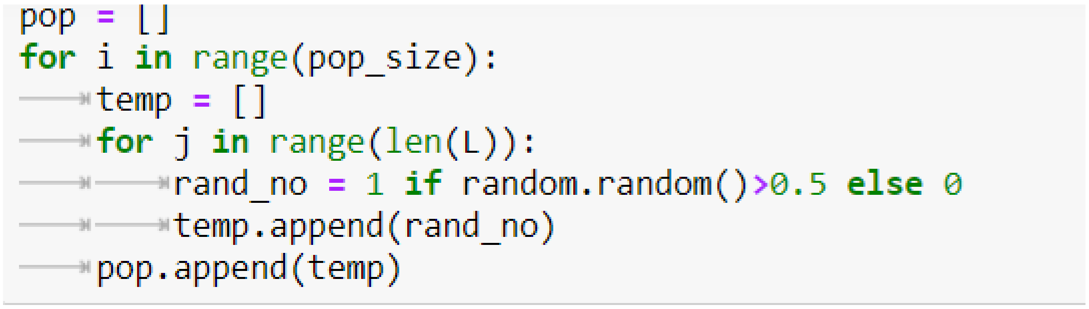
The chromosome of the GA contains all the building blocks to a solution of the problem at hand in a form that is suitable for the genetic operators and the fitness function. In genetic algorithms, a chromosome is a set of parameters or factors that determine a proposed solution to the intended problem. In our proposed model, each gene in the chromosome indicates to a candidate customer. The value of a gene can be either 1 or 0, where 1 indicates that the corresponding customer is elected to get the loan and 0 indicates a non-selected customer. Fig 1 depicts a chromosome for a field with 25 customers.

Algorithm Implementation

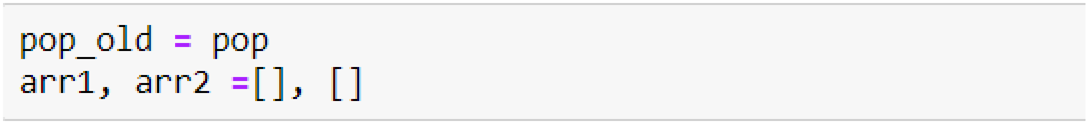
For Part 1

a) Initialisation of required parameters of the Genetic Algorithm.
b) Generation of Initial Population
c) Fitness is calculated and Constraint is checked for each chromosome and Selection is done using Roulette Wheel
d) Single Point Crossover is done followed by Mutation and a new Generation is generated.
e) c, d are repeated in a loop where we store Best Fitness and Best Individual in an array for every loop and range of the loop is the number of Generations, which is defined as 60 in our case.

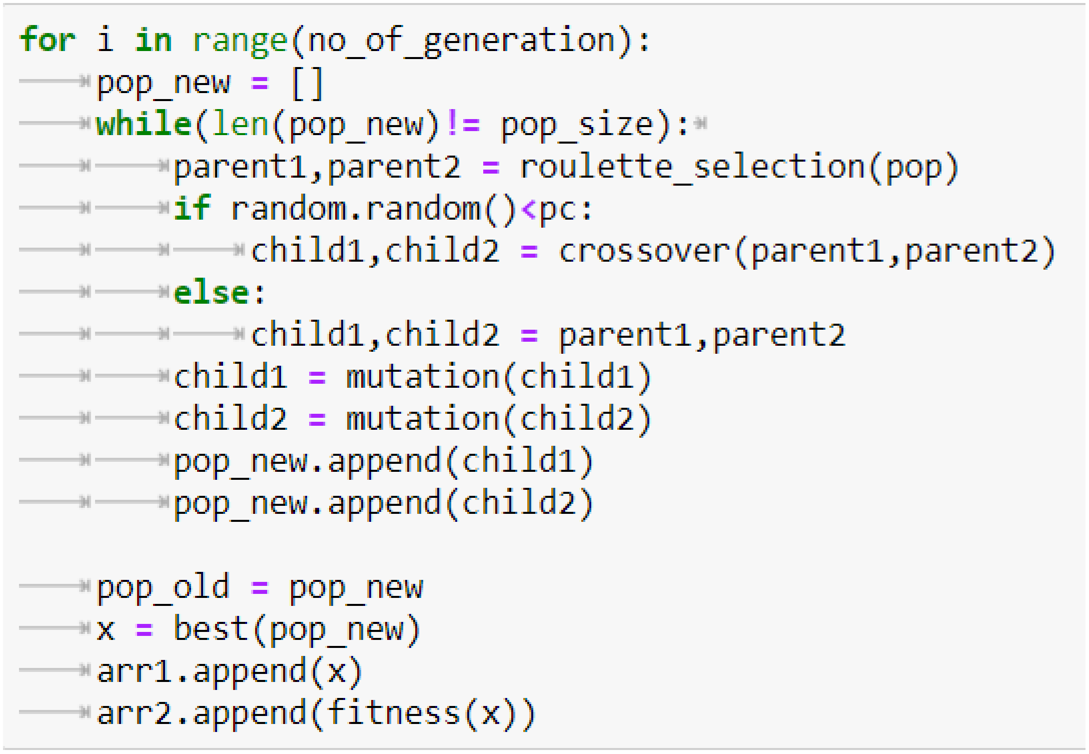
Main Code for this Algorithm is Shown Below



```
pop = []
for i in range(pop_size):
    temp = []
    for j in range(len(L)):
        rand_no = 1 if random.random()>0.5 else 0
        temp.append(rand_no)
    pop.append(temp)
```



```
pop_old = pop
arr1, arr2 =[], []
```



```
for i in range(no_of_generation):
    pop_new = []
    while(len(pop_new)!= pop_size):
        parent1,parent2 = roulette_selection(pop)
        if random.random()<pc:
            child1,child2 = crossover(parent1,parent2)
        else:
            child1,child2 = parent1,parent2
        child1 = mutation(child1)
        child2 = mutation(child2)
        pop_new.append(child1)
        pop_new.append(child2)

    pop_old = pop_new
    x = best(pop_new)
    arr1.append(x)
    arr2.append(fitness(x))
```

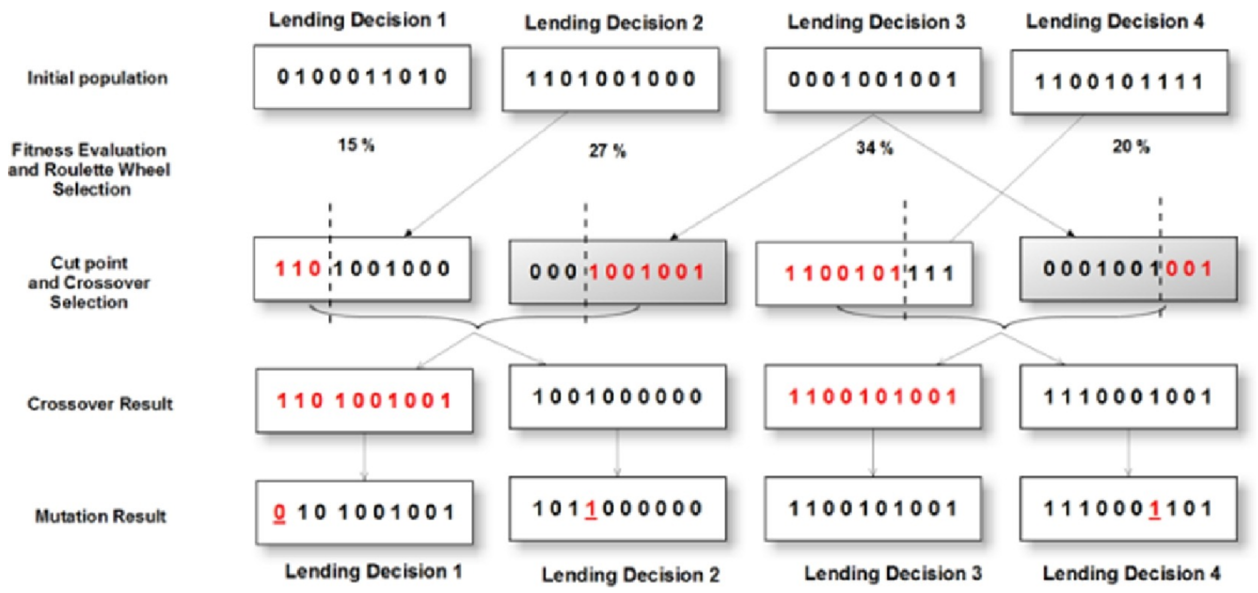
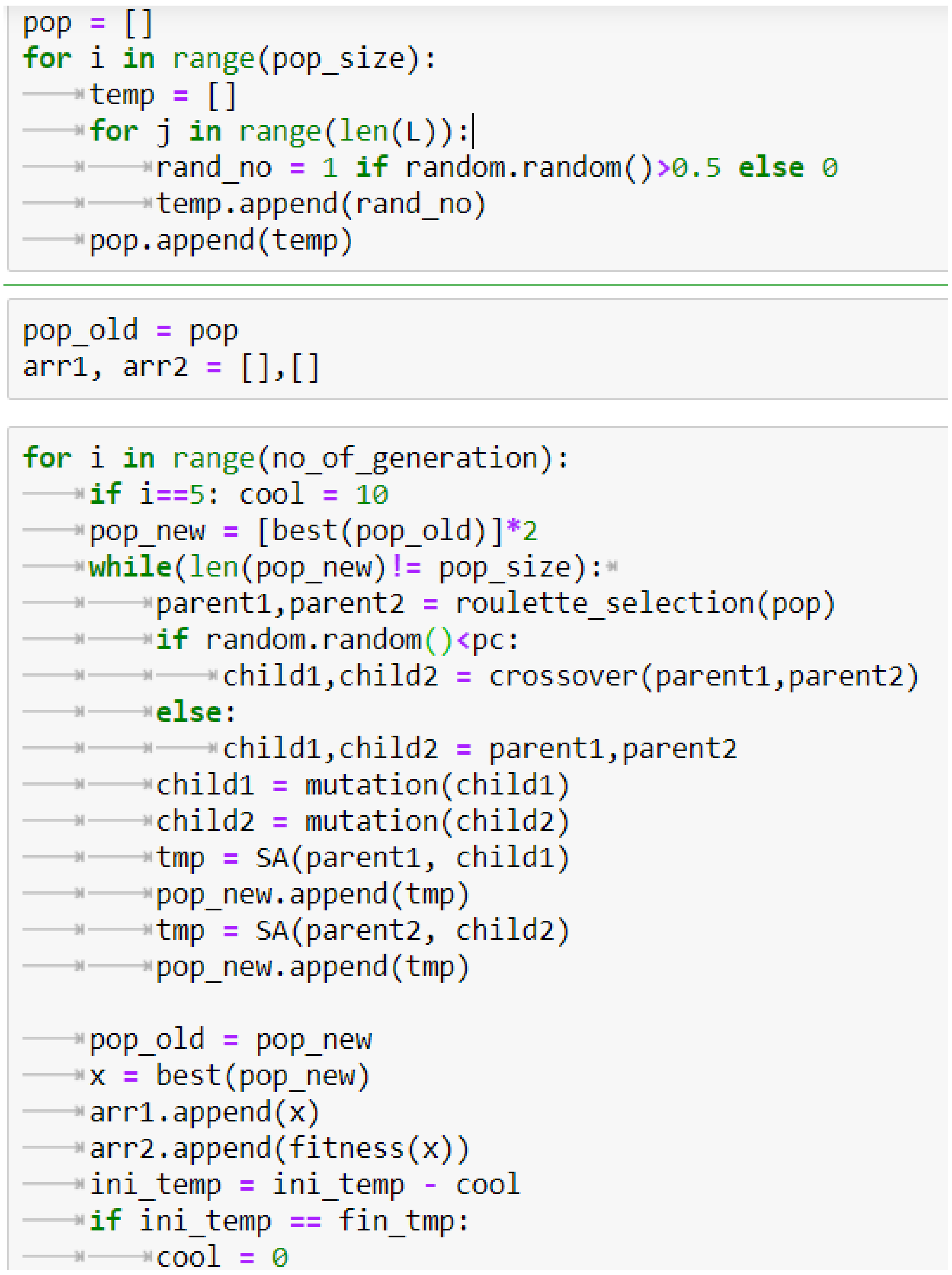


Fig. 2. An example for the GA operations - Crossover and Mutation

<u>For Part 2</u>

a) Initialisation of required parameters of the Genetic algorithm and Simulated Annealing.
b) Generation of Initial Population
c) Fitness is calculated and Constraint is checked for each chromosome and Selection is done using Roulette Wheel
d) Single Point Crossover is done followed by Mutation and a new Generation is generated.
e) If Fitness of Child is better than the Parent then we keep the Child in the new generation else we use the Simulated Annealing concept here. We check if the acceptance probability is greater than the Random Number generated, we keep the child in New Generation else we keep Parent.
f) c, d, e are repeated in a loop where we store Best Fitness and Best Individual in an array for every loop and range of the loop is the Number of Generations, which is defined as 60 in our case.

Main Code for this Algorithm is Shown Below



```python
pop = []
for i in range(pop_size):
    temp = []
    for j in range(len(L)):
        rand_no = 1 if random.random()>0.5 else 0
        temp.append(rand_no)
    pop.append(temp)
```

```python
pop_old = pop
arr1, arr2 = [],[]
```

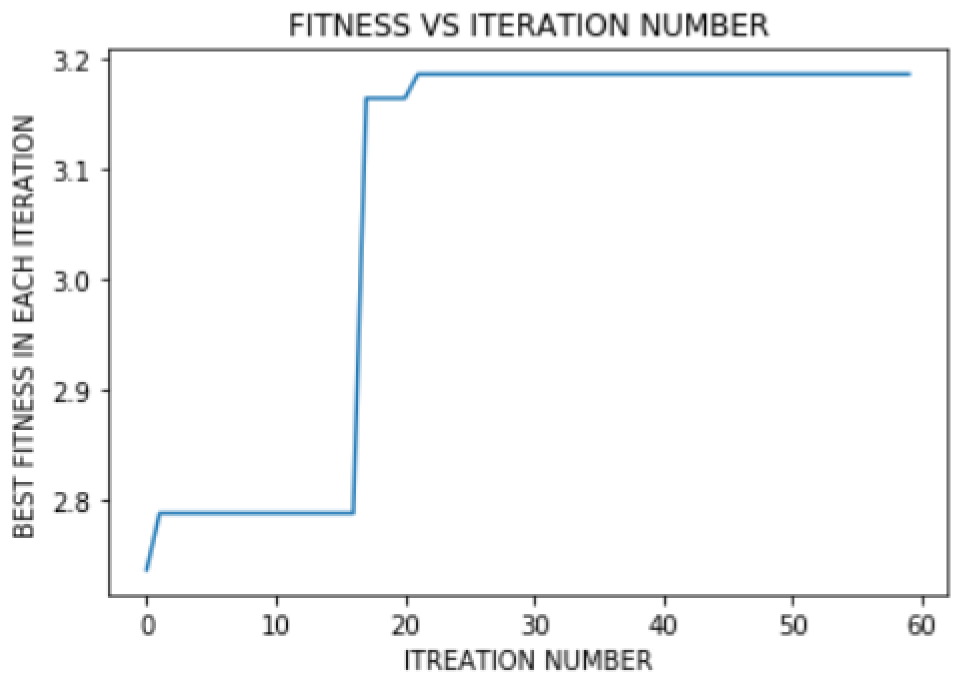
```python
for i in range(no_of_generation):
    if i==5: cool = 10
    pop_new = [best(pop_old)]*2
    while(len(pop_new)!= pop_size):
        parent1,parent2 = roulette_selection(pop)
        if random.random()<pc:
            child1,child2 = crossover(parent1,parent2)
        else:
            child1,child2 = parent1,parent2
        child1 = mutation(child1)
        child2 = mutation(child2)
        tmp = SA(parent1, child1)
        pop_new.append(tmp)
        tmp = SA(parent2, child2)
        pop_new.append(tmp)

    pop_old = pop_new
    x = best(pop_new)
    arr1.append(x)
    arr2.append(fitness(x))
    ini_temp = ini_temp - cool
    if ini_temp == fin_tmp:
        cool = 0
```

Where the SA Function is shown below

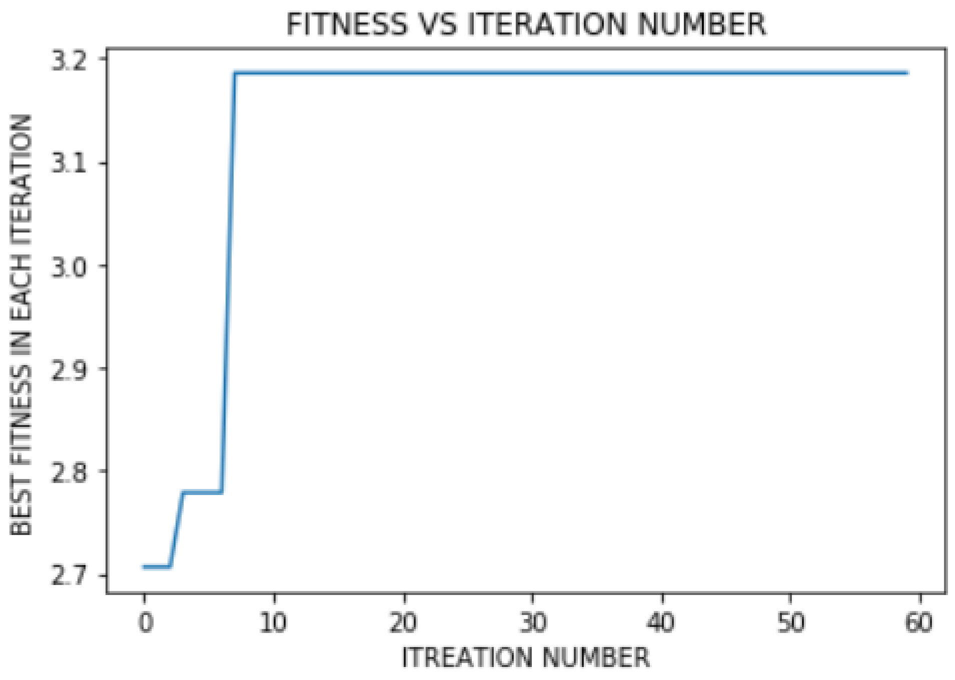
```python
def SA(parent, child):
	fchild = fitness(child)
	fparent = fitness(parent)
	if cool==0: return child
	if fchild > fparent: return child
	p = math.exp((fchild-fparent)/ini_temp)
	if random.random()<=p: return child
	return parent
```

Comparison of Both Algorithms

1) Genetic Algorithm



2) Genetic Algorithm + Simulated Annealing



RESULT ANALYSIS

1) From our results we can say that (GA + SA) Algorithm converges faster to a better solution than GA Algorithm . However the dataset which was given for this problem was very small so we can't see much difference in the results of these Algorithms.

2) Both Algorithms give the same solution. So it is fair to assume that this solution would be the optimal solution for the problem.

3) After running my code on multiple iterations I saw that the Algorithm Converges to this Best Solution mentioned below.

    Best Fitness Value = **3.1854**
    Best set of Decision variables - **1011010011**
    So our selected customers are **1,3,4,6,9,10** .

*********************** THE END ***********************