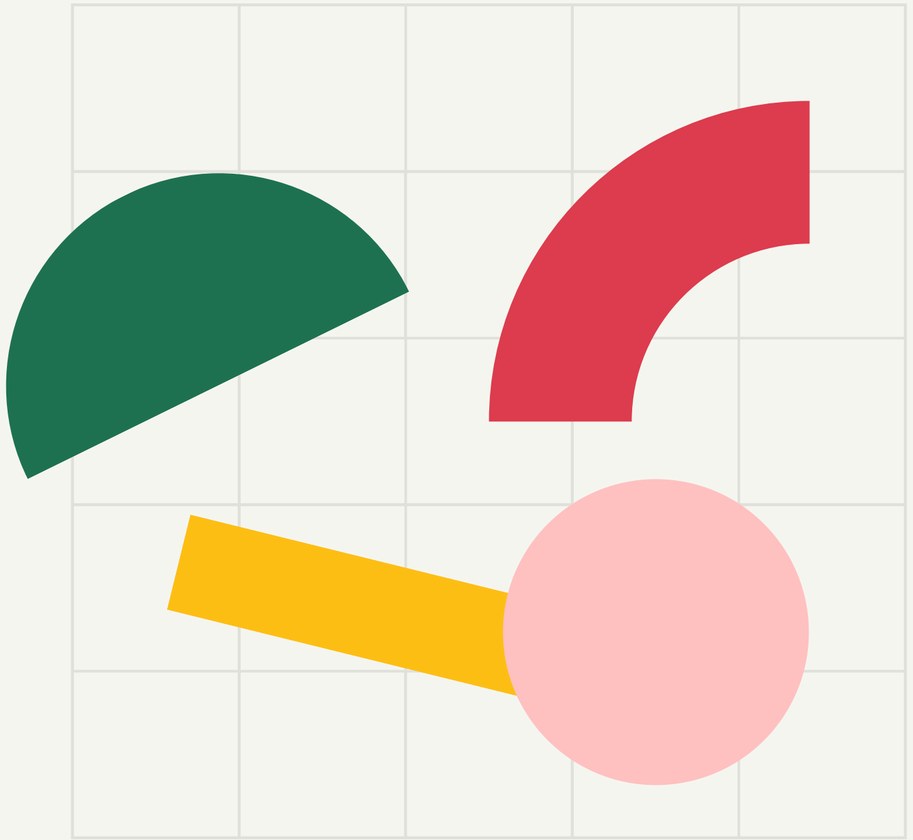
Metaheuristics Algorithms

Optimization and regression project

Contents



- Define Metaheuristics
- Problem Description
- Define Generic Algorithm
- Implement Generic Algorithm in Coding

What is metaheuristic?

Expectations and outcomes

- Strategies that guide the search process
- range from simple local search procedures to complex adaptive learning processes
- efficiently explore the search space in order to find good (near-optimal) feasible solutions
- provide no guarantee of global or local optimality
- Don't realise the unexplored feasible space
- lack a metric of goodness of solution (often stop due to an external time or iteration limit)

- are not based on some algebraic model unlike exact methods
- are often used in conjunction with an exact method
- usually are non-deterministic
- usually incorporate mechanisms to avoid getting trapped in confined areas of the search
- may use some form of memory to better guide the search
- inspired from “AI,” rather than “pure math”
- lack of theoretical “rigor” (no proofs, theorems, etc. for people to pursue)

Problem Description

Problem

A bakery wastes a quantity of flour, sugar, butter, and eggs every month
In the following quantities:

- Flour 50,00 g
- Sugar 25,00 g
- Butter 20,00 g
- Eggs 1000 g

The factory manager thought of utilizing the wasted quantities to produce other products to increase the factory profits. He is thinking of deciding about adding two additional products: biscuits and cupcakes

Knowing that the quantity needed to produce one box of biscuits is:
200 grams of flour
100 grams of sugar
100 grams of butter
20 grams of eggs
However, the quantity needed to produce one cupcake box is:
300 grams of flour
200 grams of sugar
200 grams of butter
30 grams of eggs
The price of a box of biscuits is $5, and the price of a box of cupcakes is $8

How many boxes does the factory need to produce biscuits and cupcakes to achieve the highest profits?

Problem Description

Objective function

P= 5x + 8y

Constraints

$200x + 300y \leqslant 50{,}000$
$100x + 200y \leqslant 25{,}000$
$100x + 200y \leqslant 20{,}000$
$20x + 30y \leqslant 1000$
$x, y \geqslant 0$

Mathematical representation :

Maximize P= 5x + 8y
Subject to
$2x + 3y \leqslant 500$
$x + 2y \leqslant 250$
$x + 2y \leqslant 200$
$2x + 3y \leqslant 100$
$x, y \geqslant 0$

Genetic Algorithms

Brief Introduction

Genetic algorithms (GAs) are search algorithms based on the mechanism of natural selection. It is used to find the optimal or nearest optimal solution.

Components of Genetic Algorithms

Population, Chromosome, Gene

Population

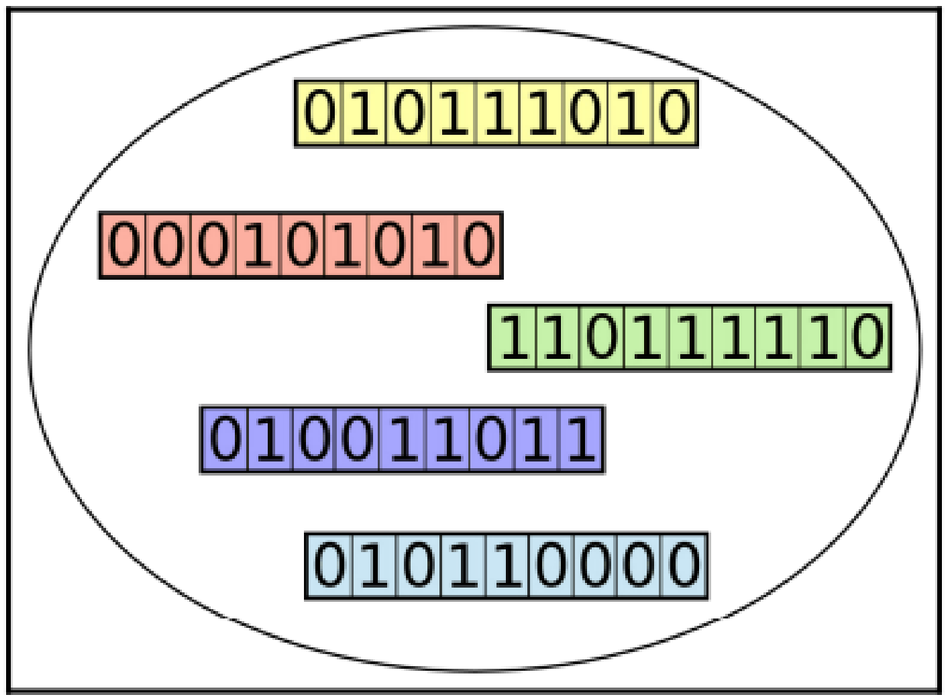
Population: a collection of candidate solutions for the problem at hand

Chromosome

Chromosome is one of the solution to that problem.

Gene

each chromosome is a set of genes.



Components of Genetic Algorithms

Fitness function

The fitness function represents the problem we would like to solve, The objective of genetic algorithms is to find the individuals that yield the highest value when this function is calculated for (also called the target function or objective function)
in our problem the Fitness function , f(x,y) = 5x + 8y
This is the function we seek to optimize it

Selection

is a process used to determine which of the individuals in the population will get to reproduce and create the offspring that will form the next generation.

Components of Genetic Algorithms

Crossover

is a operation used to create a pair of new individuals,This is usually done by taking two selected individuals at a time and interchanging parts of their chromosomes to create two new chromosomes representing the offspring

Mutation

The mutation is an operation used to bring diversity to the population. There are different kinds of mutations like Bit Flip mutation, Swap mutation, Inversion mutation, etc.

Are you ready?

Now, we are going to implement the Genetic Algorithms
To find nearly optimal solution for our problem

Implement Genetic Algorithm

Some methods help us to Implement Genetic Algorithm :

- Fitness_FunctionB(X)
- checkConstraintsB(X)
- ConvertToDecmail(pair)
- GenerateInitialPopulation(populations_size)
- sortedbest(Array)
- selectparent(Array)
- crossover(pop,Array)
- mutation(newp)

Implement Genetic Algorithm

Creating the initial population

```
## -- GenerateInitialPopulation function that takes populations_size as input
## --and initializing the population randomly, based on the problem restrictions """

def GenerateInitialPopulation(populations_size):
    Array = []
    while(len(Array) != populations_size):
        d = []
        X = 0
        Y = 0
        while True :
            X = [random.randint(0,1) for x in range(number_Of_gens)]#
            Y = [random.randint(0,1) for x in range(number_Of_gens)] #
            if checkConstraintsB([X,Y]) == True:
                break
        Array.append([X,Y])
    return Array
```

The initial population is a set of valid candidate solutions (individuals) chosen randomly.

- population size : 6
- number of bits : 6
- chromosome format : Each chromosome contains a set of variable values in binary representation

Implement Genetic Algorithm

- number of bits : 6
- population size : 6

Creating the initial population

0, 1, 0, 0, 0, 1	0, 0, 1, 1, 1, 1
1, 0, 0, 0, 0, 1	0, 0, 0, 0, 0, 0
0, 1, 1, 1, 0, 0	0, 1, 1, 1, 0, 0
1, 0, 0, 0, 1, 1	0, 0, 0, 0, 0, 1
0, 0, 0, 0, 1, 0	0, 0, 1, 0, 1, 0]
0, 1, 0, 0, 1, 0	0, 0, 1, 0, 0, 1

Chromosome

Gene

Population

Implement Genetic Algorithm

Sort The individuals , Calculating the fitness

```
## -- sortedbest function that takes an array of population as input
## -- and then sorting them based on fitness from best to worst

def sortedbest(Array):

    for i in range(populationsSize):
        j = i+1

        for j in range(populationsSize-1):
            Best = Fitness_FunctionB(Array[i])
            Fitness_value = Fitness_FunctionB(Array[j])

            if(Best>Fitness_value):
                index = i
                new = Array[i]
                Array[i] = Array[j]
                Array[j] = new

    return Array.copy()
```

Implement Genetic Algorithm

Sort The individuals , Calculating the fitness

```
## -- The Fitness_FunctionB
## -- calculates the fitness value of single posibble solution (x,y)
def Fitness_FunctionB(X):

    ## Calling the ConvertToDecmail function
    # to convert X from binary to decimal
    decimalPair = ConvertToDecmail(X)
    x, y = decimalPair[0] , decimalPair[1]

    return 5*x + 8*y
```

The value of the fitness function is calculated for each individual. This is done once for the initial population, and then for every new generation after applying the genetic operators of selection, crossover, and mutation

Creating the initial population

initial population in binary

0, 1, 0, 0, 0, 1	0, 0, 1, 1, 1, 1
1, 0, 0, 0, 0, 1	0, 0, 0, 0, 0, 0
0, 1, 1, 1, 0, 0	0, 0, 0, 0, 1, 0
1, 0, 0, 0, 1, 1	0, 0, 0, 0, 0, 1
0, 0, 0, 0, 1, 0	0, 0, 1, 0, 1, 0
0, 1, 0, 0, 1, 0	0, 0, 1, 0, 0, 1

Creating the initial population

initial population in binary		After convert individuals to decimal	
0, 1, 0, 0, 0, 1	0, 0, 1, 1, 1, 1	17	15
1, 0, 0, 0, 0, 1	0, 0, 0, 0, 0, 0	33	0
0, 1, 1, 1, 0, 0	0, 0, 0, 0, 1, 0	28	2
1, 0, 0, 0, 1, 1	0, 0, 0, 0, 0, 1	35	1
0, 0, 0, 0, 1, 0	0, 0, 1, 0, 1, 0	2	10
0, 1, 0, 0, 1, 0	0, 0, 1, 0, 0, 1	18	9

Creating the initial population

initial population in binary		After convert individuals to decimal		fitness
0, 1, 0, 0, 0, 1	0, 0, 1, 1, 1, 1	17	15	205
1, 0, 0, 0, 0, 1	0, 0, 0, 0, 0, 0	33	0	165
0, 1, 1, 1, 0, 0	0, 0, 0, 0, 1, 0	28	2	156
1, 0, 0, 0, 1, 1	0, 0, 0, 0, 0, 1	35	1	183
0, 0, 0, 0, 1, 0	0, 0, 1, 0, 1, 0	2	10	90
0, 1, 0, 0, 1, 0	0, 0, 1, 0, 0, 1	18	9	162

Creating the initial population

Initial population after sorting

0, 1, 0, 0, 0, 1	0, 0, 1, 1, 1, 1
1, 0, 0, 0, 1, 1	0, 0, 0, 0, 0, 1
1, 0, 0, 0, 0, 1	0, 0, 0, 0, 0, 0
0, 1, 0, 0, 1, 0	0, 0, 1, 0, 0, 1
0, 1, 1, 1, 0, 0	0, 0, 0, 0, 1, 0
0, 0, 0, 0, 1, 0	0, 0, 1, 0, 1, 0

Implement Genetic Algorithm

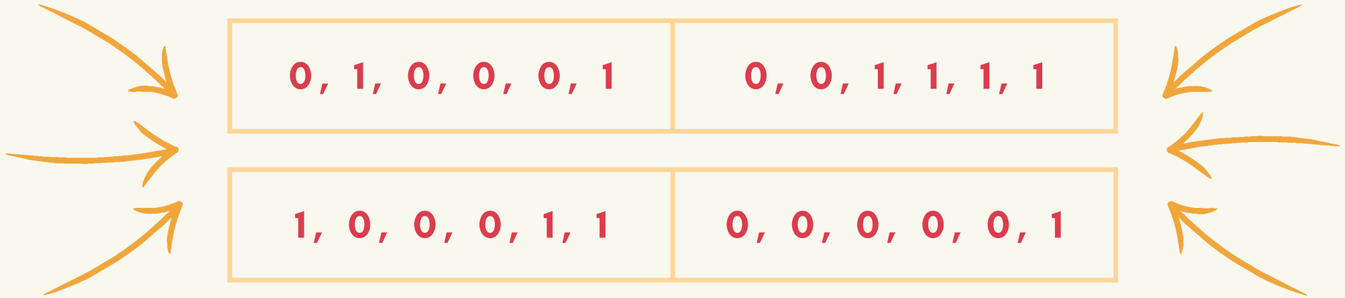
Applying selection

```
# selectparent function takes a array of population as input
# then arrange all posibble sol. by calling sortedbest function
# and then selecting the fittest pairs of (x,y)
#are selected based on the fitness values. --process of Selecting Parents

def selectparent(Array):

    p = sortedbest(Array)
    return p[0:2]
```

The selection operator is responsible for selecting individuals from the current population in a way that gives an advantage to better individuals.

Selection Operator

selecting parents



0, 1, 0, 0, 0, 1	0, 0, 1, 1, 1, 1
1, 0, 0, 0, 1, 1	0, 0, 0, 0, 0, 1
1, 0, 0, 0, 0, 1	0, 0, 0, 0, 0, 0
0, 1, 0, 0, 1, 0	0, 0, 1, 0, 0, 1
0, 1, 1, 1, 0, 0	0, 0, 0, 0, 1, 0
0, 0, 0, 0, 1, 0	0, 0, 1, 0, 1, 0

Implement Genetic Algorithm

Applying crossover

The crossover (or recombination) operator creates offspring from the selected parents and other individuals. This is usually done by taking two selected individuals at a time and interchanging parts of their chromosomes to create two new chromosomes representing the offspring.

Implement Genetic Algorithm

Applying crossover

```
##-- crossover function takes Population and parents as input
##--creates offspring from the selected individuals.
##-- a crossover function is required to generate offsprings.

def crossover(pop,Array) :
    newPop  = Array.copy() # Store parents in the new population

    cross_point = random.randint(1,5)
    X = (Array[0][0][0:cross_point] +Array[1][0][cross_point:number_Of_gens])
    Y = (Array[0][1][0:cross_point] +Array[1][1][cross_point:number_Of_gens])

    while(checkConstraintsB([X,Y])== False):

        cross_point1 = random.randint(2,6)-1
        X = (Array[0][0][0:cross_point1] +Array[1][0][cross_point1:number_Of_gens])
        Y = (Array[1][1][0:cross_point1] +Array[1][1][cross_point1:number_Of_gens])
        newPop.append([X,Y])

    while(len(newPop) != len(pop)):
        cross_point = random.randint(0,5)
        x = random.randint(0,5)
        y = random.randint(0,1)
        u = 0 if y == 1 else 1
        X = (pop[x][y][0:cross_point] +pop[x][y][cross_point:number_Of_gens])
        Y = (pop[x][u][0:cross_point] +pop[x][u][cross_point:number_Of_gens])

        while(checkConstraintsB([X,Y])== False):
            cross_point1 = random.randint(0,5)
            x = random.randint(0,5)
            y = random.randint(0,1)
            u = 0 if y == 1 else 1
            X = (pop[x][y][0:cross_point] +pop[x][y][cross_point:number_Of_gens])
            Y = (pop[x][u][0:cross_point] +pop[x][u][cross_point:number_Of_gens])
        newPop.append([X,Y])
    newPop = sortedbest(newPop)

    return newPop
```

Create Off-Spring crossover

0, 1, 0, 0, 0, 1	0, 0, 1, 1, 1, 1
1, 0, 0, 0, 1, 1	0, 0, 0, 0, 0, 1
1, 0, 0, 0, 0, 1	0, 0, 0, 0, 0, 0
0, 1, 0, 0, 1, 0	0, 0, 1, 0, 0, 1
0, 1, 1, 1, 0, 0	0, 0, 0, 0, 1, 0
0, 0, 0, 0, 1, 0	0, 0, 1, 0, 1, 0

Cross point = 2

0, 1,0, 0, 1, 1	0, 0,0, 0, 0, 1

Create Off-Spring crossover

0, 1, 0, 0, 0, 1	0, 0, 1, 1, 1, 1
1, 0, 0, 0, 1, 1	0, 0, 0, 0, 0, 1
1, 0, 0, 0, 0, 1	0, 0, 0, 0, 0, 0
0, 1, 0, 0, 1, 0	0, 0, 1, 0, 0, 1
0, 1, 1, 1, 0, 0	0, 0, 0, 0, 1, 0
0, 0, 0, 0, 1, 0	0, 0, 1, 0, 1, 0

Cross point = 2

0, 1,0, 0, 1, 1	0, 0,0, 0, 0, 1

Cross point = 3

1, 0, 0 ,0, 1, 0	0, 0,0, 0, 0, 1

Create Off-Spring crossover

0, 1, 0, 0, 0, 1	0, 0, 1, 1, 1, 1
1, 0, 0, 0, 1, 1	0, 0, 0, 0, 0, 1
1, 0, 0, 0, 0, 1	0, 0, 0, 0, 0, 0
0, 1, 0, 0, 1, 0	0, 0, 1, 0, 0, 1
0, 1, 1, 1, 0, 0	0, 0, 0, 0, 1, 0
0, 0, 0, 0, 1, 0	0, 0, 1, 0, 1, 0

Cross point = 2

0, 1,0, 0, 1, 1	0, 0,0, 0, 0, 1

✓

Cross point = 3

1, 0, 0 ,0, 1, 0	0, 0,0, 0, 0, 1

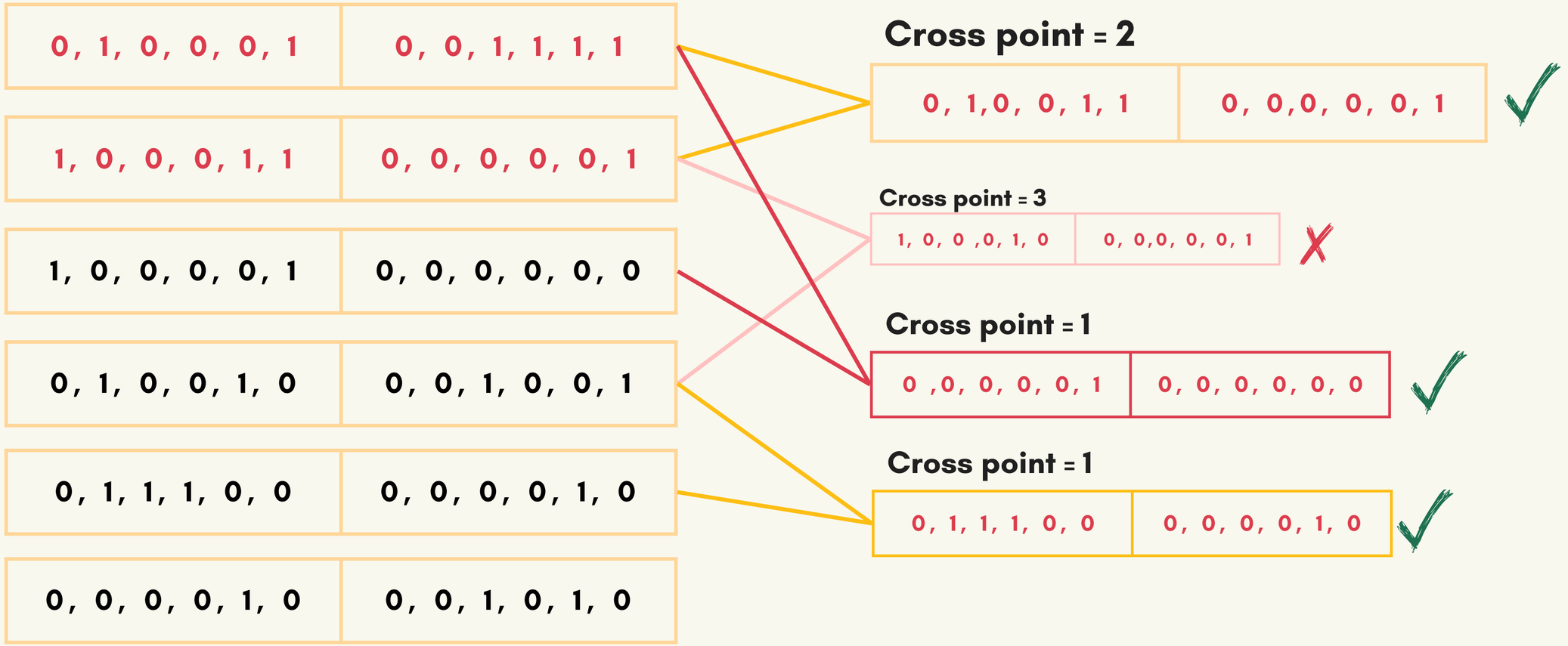
✗

Cross point = 1

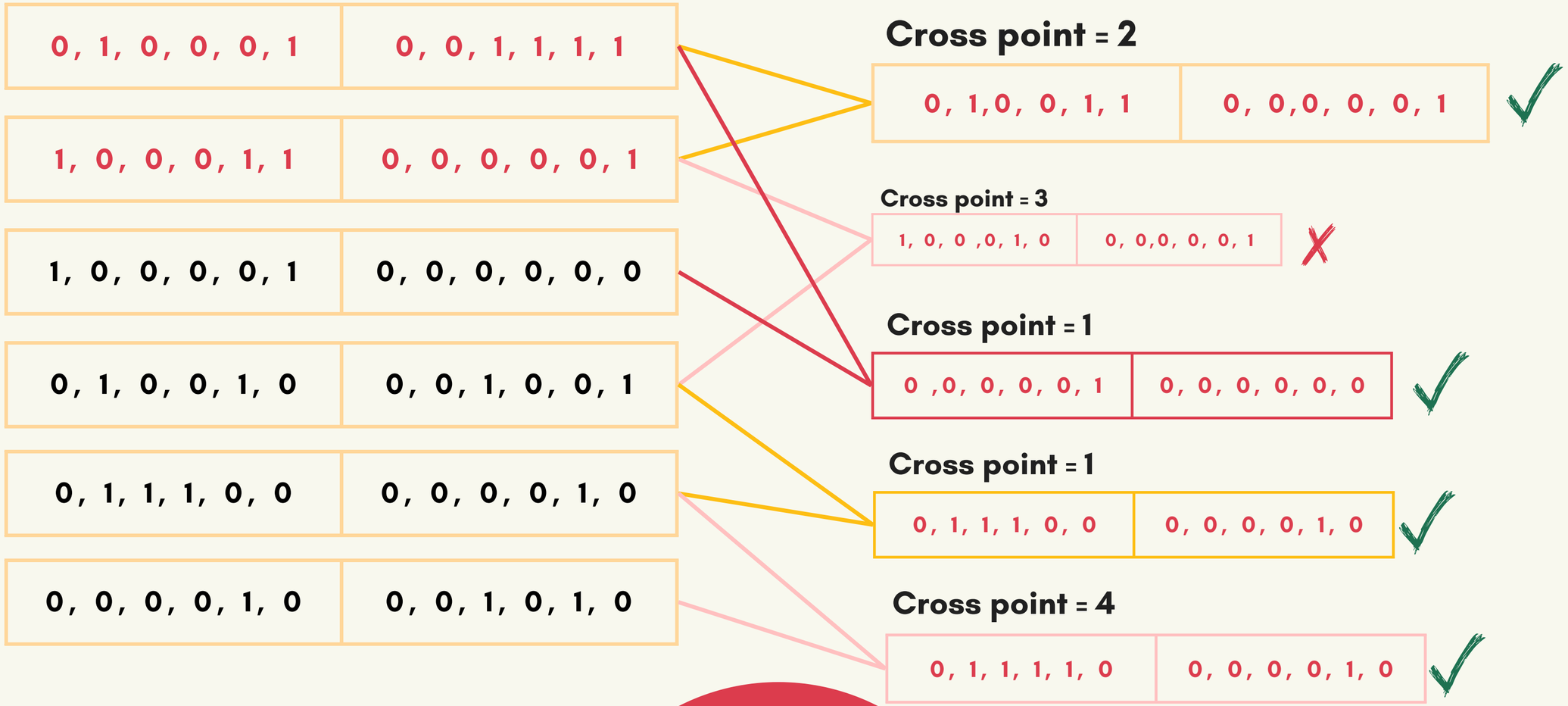
0 ,0, 0, 0, 0, 1	0, 0, 0, 0, 0, 0

✓

Create Off-Spring crossover



Create Off-Spring crossover



Create Off-Spring crossover

After crossover

0, 1, 0, 0, 0, 1	0, 0, 1, 1, 1, 1
1, 0, 0, 0, 1, 1	0, 0, 0, 0, 0, 1
0, 1, 1, 1, 1, 0	0, 0, 0, 0, 1, 0
0, 1, 1, 1, 0, 0	0, 0, 0, 0, 1, 0
0 ,0, 0, 0, 0, 1	0, 0, 0, 0, 0, 0
0, 1,0, 0, 1, 1	0, 0,0, 0, 0, 1

Applying Mutation

```
## -- mutation function takes a population after crossover opretor as input
## -- and then randomly introduce a change to one of (x,y) values by change a value of bit (gene)
def mutation(newp) :

    x = random.randint(0,5)
    Z = random.randint(4,5)
    newp[x][0][Z] = 0 if newp[x][0][Z] == 1 else newp[x][0][Z]
    newp[x][1][Z] = 0 if newp[x][1][Z] == 1 else newp[x][1][Z]
    newp = sortedbest(newp)
    return newp
```

The mutation operator can randomly introduce a change to one or more of the chromosome values (genes) of each newly created individual.

Applying Mutation

Select one individual randomly

0, 1, 0, 0, 0, 1	0, 0, 1, 1, 1, 1
1, 0, 0, 0, 1, 1	0, 0, 0, 0, 0, 1
0, 1, 1, 1, 1, 0	0, 0, 0, 0, 1, 0
0, 1, 1, 1, 0, 0	0, 0, 0, 0, 1, 0
0 ,0, 0, 0, 0, 1	0, 0, 0, 0, 0, 0
0, 1,0, 0, 1, 1	0, 0,0, 0, 0, 1

Applying Mutation

Select index of bit randomly

0, 1, 0, 0, 0, 1	0, 0, 1, 1, 1, 1
1, 0, 0, 0, 1, 1	0, 0, 0, 0, 0, 1
0, 1, 1, 1, 1, 0	0, 0, 0, 0, 1, 0
0, 1, 1, 1, 0, 0	0, 0, 0, 0, 1, 0
0 ,0, 0, 0, 0, 1	0, 0, 0, 0, 0, 0
0, 1,0, 0, 1, 1	0, 0,0, 0, 0, 1

Applying Mutation

Flipping randomly

0, 1, 0, 0, 0, 1	0, 0, 1, 1, 1, 1
1, 0, 0, 0, 1, 1	0, 0, 0, 0, 0, 1
0, 1, 1, 1, 0, 0	0, 0, 0, 0, 0, 0
0, 1, 1, 1, 0, 0	0, 0, 0, 0, 1, 0
0 ,0, 0, 0, 0, 1	0, 0, 0, 0, 0, 0
0, 1,0, 0, 1, 1	0, 0,0, 0, 0, 1

Implement Genetic Algorithm

```
numberOfIterations = 300 # defined number of iterations

Binary_populations = GenerateInitialPopulation(populationsSize)
for i in range(numberOfIterations):
    parents = selectparent(Binary_populations)
    new_populations = crossover(Binary_populations,parents)
    Binary_populations = mutation(new_populations)
```

```
for i in range(6):
    d = ConvertToDecmail(Binary_populations[i])
    print("x = ", d[0] , " y = ", d[1] , "with Fitness value = " ,Fitness_FunctionB(Binary_populations[i]))
    Message = "Yes" if checkConstraintsB(Binary_populations[i]) == True else "No !!"
    print("Does this solution check for restrictions  " , Message , "\n")
```

```
print("The best solution is \nIn binary representation:" , Binary_populations[0])
print("In Decmail representation:" ,ConvertToDecmail(Binary_populations[0]))
print("----Fitness value = " , Fitness_FunctionB(Binary_populations[0]))
```

Different running outputs

Here are outputs (solutions) in different running times

Running 1

```
x =  45  y =  3 with Fitness value =  249
Does this solution check for restrictions   Yes

x =  44  y =  2 with Fitness value =  236
Does this solution check for restrictions   Yes

x =  31  y =  5 with Fitness value =  195
Does this solution check for restrictions   Yes

x =  16  y =  14 with Fitness value =  192
Does this solution check for restrictions   Yes

x =  1  y =  23 with Fitness value =  189
Does this solution check for restrictions   Yes

x =  33  y =  1 with Fitness value =  173
Does this solution check for restrictions   Yes
```

```
The best solution is
In binary representation: [[1, 0, 1, 1, 0, 1], [0, 0, 0, 0, 1, 1]]
In Decmail representation: [45, 3]
----Fitness value =  249
```

Different running outputs

Here are outputs (solutions) in different running times

Running 2

```
x =  0  y =  32 with Fitness value =  256
Does this solution check for restrictions   Yes

x =  12  y =  16 with Fitness value =  188
Does this solution check for restrictions   Yes

x =  12  y =  16 with Fitness value =  188
Does this solution check for restrictions   Yes

x =  16  y =  12 with Fitness value =  176
Does this solution check for restrictions   Yes

x =  32  y =  0 with Fitness value =  160
Does this solution check for restrictions   Yes

x =  32  y =  0 with Fitness value =  160
Does this solution check for restrictions   Yes
```

```
The best solution is
In binary representation: [[0, 0, 0, 0, 0, 0], [1, 0, 0, 0, 0, 0]]
In Decmail representation: [0, 32]
----Fitness value =  256
```

Different running outputs

Here are outputs (solutions) in different running times

Running 3

```
x =  0  y =  33 with Fitness value =  264
Does this solution check for restrictions   Yes

x =  0  y =  33 with Fitness value =  264
Does this solution check for restrictions   Yes

x =  33  y =  0 with Fitness value =  165
Does this solution check for restrictions   Yes

x =  33  y =  0 with Fitness value =  165
Does this solution check for restrictions   Yes

x =  32  y =  0 with Fitness value =  160
Does this solution check for restrictions   Yes

x =  32  y =  0 with Fitness value =  160
Does this solution check for restrictions   Yes
```

```
The best solution is
In binary representation: [[0, 0, 0, 0, 0, 0], [1, 0, 0, 0, 0, 1]]
In Decmail representation: [0, 33]
----Fitness value =  264
```

Thank you