CII2M3-IF-44-INT - INTRODUCTION TO ARTIFICIAL INTELLIGENCE

EVEN SEMESTER SESSION 2021/2022

PROGRAMMING ASSIGNMENT 1- SEARCHING

Group No : 5

Section: IF-44-INT

Lecturer Name: SIR EDWARD FERDIAN

Group Member:

NAME	STUDENT ID
MUHAMMAD FADLI RAMADHAN	1301203533
NUR FASIHAH AYUNI BINTI MOHD YAHYA	1301213670

Programming Assignment 01 - Searching

1.0 - Question

Analyze and design Genetic Algorithm (GA) and implement a program to find x dan y values to obtain the minimum value from the following function

$$h(x,y) = \frac{(\cos x + \sin y)^2}{x^2 + y^2}$$

with the following **domain** for x and y:

$$-5 \leq x \leq 5 \text{ dan } -5 \leq y \leq 5$$

2.0 - Problem solving

Genetic Algorithm

What is a genetic algorithm? Genetic Algorithm is a particular class of evolutionary algorithm that uses techniques inspired by evolutionary biology such as inheritance, mutation, selection, and crossover. Genetic algorithm is also a search heuristic that is inspired by Charles Darwin's theory. To solve the problem, we are using Genetic Algorithm by implementing a program by using Python Language.

In this report, we are going to explain how the genetic algorithm(GA) works by solving the problem. Any optimization problem starts with an objective function. The function that been given is :

$$h(x,y) = \frac{(\cos x + \sin y)^2}{x^2 + y^2}$$

There is a flowchart Basic Structure of Genetic Algorithm in our problem :

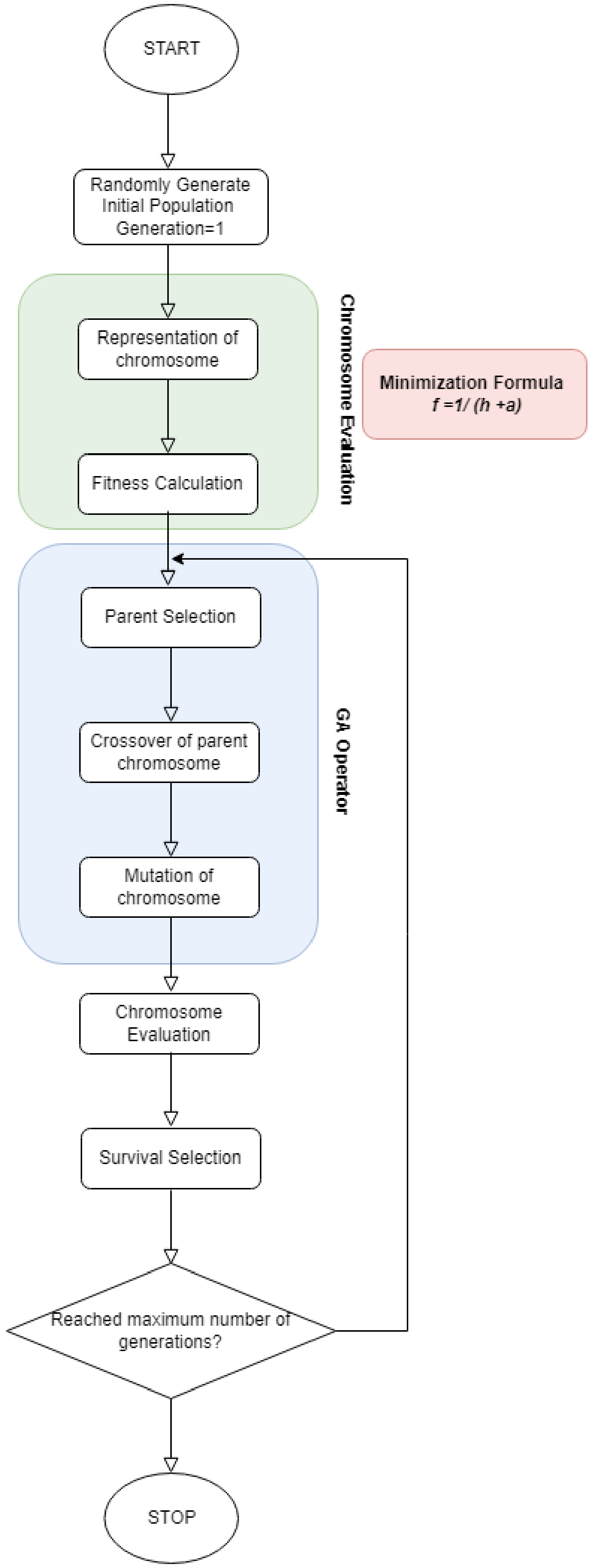


Figure 1 Basic Structure of Genetic Algorithm

STEP 1- Initialize Population

First, this step starts with sets of values called 'chromosomes' and the step is called 'initialize population'. Here population means set of x and y [x,y]. Random choices function is used to generate the initial values of x and y. Usually, binary values are used to generate the value (string of 1s and 0s)). As we can see in the coding we are using binary values which are 0 and 1 to generate the initial population. If the population is None so it will generate the initial population

Example of initialization in our coding:

```
# GA parameter
dmin_x = -5   # Minimum value of X
dmax_x = 5    # Maximum value of x
dmin_y = -5   # Minimum value of y
dmax_y = 5    # Maximum value of y
```

```
def __init__(self, bin = None):
    if bin == None:
        b=[0,1]
        self.bin = random.choices(b, k=6)
```

STEP 2- Phenotype/ Decode

Next, after generating random initial population we append(add) the value to the representation function. In individual representation, we use the formula of Binary Encoding using 6 bits (6 gens) in which every chromosome is a string of bits, 0 or 1 represent a particular characteristic of the problem. In this step we split the 2 chromosome from the 6 gen using this formula:

$$x = r_{min} + \frac{r_{max} - r_{min}}{\sum_{i=1}^{N} 2^{-i}} (g_1 * 2^{-1} + g_2 * 2^{-2} + \cdots + g_N * 2^{-N})$$

Figure 2 Binary Encoding Formula

Example of decode in our coding:

```
# using binary encoding
    def decode(self, dmax, dmin, g):
        tp = [2**-i for i in range(1, len(g) + 1)]
        return dmin + ((dmax - dmin) / sum(tp) * sum([g[i] * tp[i] for i in range(len(g))]))
```

```
else:
    self.bin = bin
self.x = self.decode(dmin_x, dmax_x, self.bin[:3])
self.y = self.decode(dmin_y, dmax_y, self.bin[3:])
```

STEP 3 - Calculation Fitness

In this step, we calculate the value from the phenotype by using the function or formula in the question :

$$h(x, y) = \frac{(\cos x + \sin y)^2}{x^2 + y^2}$$

Figure 3 Function given in the question

Example of heuristic value in our coding :

```
# Heuristic Value
def HeuristicValue (x,y):
    return  (((math.cos(x) + math.sin(y))*(math.cos(x) + math.sin(y)))/((x*x) + (y*y) ))# Formula of heuristic value
```

After we get the value of h(x,y), we calculate the fitness by using the minimization formula **F= 1/h(x,y)+a.** Then we produce the output from the calculation fitness. As we can see in the coding we put the value of h(x/y) as h and 0.00000000000000000000000001 as a

Example of fitness value in our coding:

```
def fitness_value (x,y):   # Minimization Heuristic
    return 1/(((math.cos(x) + math.sin(y))*(math.cos(x) + math.sin(y)))/((x*x) + (y*y) + 0.00000000000000000000000001))
```

STEP 4- Parent Selection

Besides, this process is to select individuals as parents to generate new offspring for the next generation. We are using the method for parent selection is **Roulette Wheel Selection**. If fi is the fitness of individual i in the population, its probability of being selected is :

$$p_i = \frac{f_i}{\Sigma_{j=1}^{N} f_j}$$

Figure 4 Roulette Wheel Selection

As we can see in the coding, we also create the condition that it will not choose the same parent within 2 chromosomes.

Example of parent selection in our coding:

```
# Parent Selection using method Roullete Wheel Selection
def roulleteWheel(k):
    parent=[] #create an array/list for parent

    # create lambda function as an anonymous function inside other function to sort
    fitness_list = list(map(lambda ch: fitness_value (ch.x , ch.y), Population))
    w_list = [fitness_list[i] / sum (fitness_list) for i in range (len (Population))]

    while len(parent) != k:
        select_p = random.choices(Population, weights=w_list)[0]
        if not found (parent,select_p): # not have same chromosome within 2 parents
            parent.append(select_p)
    return parent
```

STEP 5- Crossover

In this step we called **'crossover'.** Crossover is 'the change of a single (0,1) or a group of genes [1,0,1]' occurred because of mating between two parent chromosomes. The crossover will randomly select a point between the parent1 and parent2 and cross the point. The operation is called 'offspring' after producing new chromosomes which are Child 1 and Child 2 after the crossover.

Example of crossover in our coding:

```python
def crossover (parent1, parent2):
# Randomly choose 1 cutting point
    cuttingpoint= random.randint(1, len(parent1.bin) - 1)

# The parent 1 and parent 2 will randomly cross between one cutting point
    Child1 = parent1.bin[:cuttingpoint] + parent2.bin[cuttingpoint:]
    Child2 = parent1.bin[:cuttingpoint] + parent2.bin[cuttingpoint:]
```

STEP 6- Mutation

Moreover, this step is called **'mutation'** which is the process of altering the value from 2 individu which is child 1 and child 2 by replacing the value 1 with 0 and value 0 with 1. It depends on the mutation probability to mutate. After we get the mutation value from Child 1 and Child 2 we append the value to Population.

Example in our coding mutation in Child 1 and Child 2:

```python
# Mutation Child 1 and Child 2
    Mutation = random.uniform (0,9) #choose random value from 0 to 9
# Probability mutation < 0.4
    if Mutation < 0.4:
        Mutation_size = random.randint(0, len(Child1)-1)
        if (Child1[Mutation_size] == 0 and Child2[Mutation_size] == 0):
            Child1[Mutation_size] = 1
            Child2[Mutation_size] = 1
        elif (Child1[Mutation_size] == 1 and Child2[Mutation_size] == 1):
            Child1[Mutation_size]= 0
            Child2[Mutation_size]= 0
        elif (Child1[Mutation_size] == 0 ):
            Child2[Mutation_size] = 1
        elif (Child1[Mutation_size] == 1 ):
            Child2[Mutation_size] = 0
        elif (Child2[Mutation_size] == 0 ):
            Child1[Mutation_size] = 1
        else:
            Child2[Mutation_size] =0
```

```python
# Append the result crossover and mutation in population
    Population.append(Chromosome(Child1))
    Population.append(Chromosome(Child2))
```

STEP 7- Selection Survival

In this step, we choose the method **Steady-State Procedure** for Selection Survival. In our coding, we sort the population value by using lambda based on the Heuristic Value. In this Best Chromosome() function we remove the worst chromosome by using the 'pop' function based on fitness value. It will produce the same population . The output is the best chromosome.

Example of selection survival and Best Chromosome in our coding:

```
def selection_survivor():
    Population.sort(key= lambda ch: HeuristicValue (ch.x , ch.y), reverse = True)

def BestChromosome():
    while len(Population) != 50: # remove the worst chromosome by using pop function
        Population.pop()
```

3.0 - Screenshot of the coding

```
newGA.py ×
Desktop > C C++ > .vscode > newGA.py > roulleteWheel
import random
import math
import matplotlib.pyplot as plt

class Chromosome:
# Initialization
    def __init__(self, bin = None):
        if bin == None:
            b=[0,1]
            self.bin = random.choices(b, k=6)
        else:
            self.bin = bin
        self.x = self.decode(dmin_x, dmax_x, self.bin[:3])
        self.y = self.decode(dmin_y, dmax_y, self.bin[3:])
# output
    def __repr__(self):
        return '{} min(x,y): ({}, {})  Heuristic value : {}  Fitness value : {}'.format(self.bin, self.x, self.y, HeuristicValue(self.x,
# using binary encoding
    def decode(self, dmax, dmin, g):
        tp = [2**-i for i in range(1, len(g) + 1)]
        return dmin + ((dmax - dmin) / sum(tp) * sum([g[i] * tp[i] for i in range(len(g))]))

def found(a, ch):
    choose = False
    for i in a:
        if i.bin == ch.bin:
            choose = False
            break
    return choose

def fitness_value (x,y):   # Minimization Heuristic
    return 1/(((math.cos(x) + math.sin(y))*(math.cos(x) + math.sin(y)))/((x*x) + (y*y) + 0.0000000000000000000000001))
```

newGA.py ×

Desktop > C C++ > .vscode > newGA.py > crossover

```python
# Parent Selection using method Roullete Wheel Selection
def roulleteWheel(k):
    parent=[] #create an array/list for parent

    # create lambda function as an anonymous function inside other function to sort
    fitness_list = list(map(lambda ch: fitness_value (ch.x , ch.y), Population))
    w_list = [fitness_list[i] / sum (fitness_list) for i in range (len (Population))]

    while len(parent) != k:
        select_p = random.choices(Population, weights=w_list)[0]
        if not found (parent,select_p): # not have same chromosome within 2 parents
            parent.append(select_p)
    return parent

def crossover (parent1, parent2):
# Randomly choose 1 cutting point
    cuttingpoint= random.randint(1, len(parent1.bin) - 1)

# The parent 1 and parent 2 will randomly cross between one cutting point
    Child1 = parent1.bin[:cuttingpoint] + parent2.bin[cuttingpoint:]
    Child2 = parent1.bin[:cuttingpoint] + parent2.bin[cuttingpoint:]
```

newGA.py × Release Notes: 1.66.0

Desktop > C C++ > .vscode > newGA.py > crossover

```python
# Mutation Child 1 and Child 2
    Mutation = random.uniform (0,9) #choose random value from 0 to 9
# Probability mutation < 0.4
    if Mutation < 0.4:
        Mutation_size = random.randint(0, len(Child1)-1)
        if (Child1[Mutation_size] == 0 and Child2[Mutation_size] == 0):
            Child1[Mutation_size] = 1
            Child2[Mutation_size] = 1
        elif (Child1[Mutation_size] == 1 and Child2[Mutation_size] == 1):
            Child1[Mutation_size]= 0
            Child2[Mutation_size]= 0
        elif (Child1[Mutation_size] == 0 ):
            Child2[Mutation_size] = 1
        elif (Child1[Mutation_size] == 1 ):
            Child2[Mutation_size] = 0
        elif (Child2[Mutation_size] == 0 ):
            Child1[Mutation_size] = 1
        else:
            Child2[Mutation_size] =0

```

```python
# Append the result crossover and mutation in population
    Population.append(Chromosome(Child2))

def selection_survivor():
    Population.sort(key= lambda ch: HeuristicValue (ch.x , ch.y), reverse = True)

def BestChromosome():
    while len(Population) != 50: # remove the worst chromosome by using pop function
        Population.pop()

# Main Function

# GA parameter
dmin_x = -5   # Minimum value of X
dmax_x = 5    # Maximum value of x
dmin_y = -5   # Minimum value of y
dmax_y = 5    # Maximum value of y

# Heuristic Value
def HeuristicValue (x,y):
    return  (((math.cos(x) + math.sin(y))*(math.cos(x) + math.sin(y)))/((x*x) + (y*y) ))# Formula of heuristic value

Generation = 1
Population = []
```

```python
# Population = 50
while len(Population) != 50:
    ch = Chromosome()

    if not found(Population, ch):
        Population.append(ch)

selection_survivor()
BestChromosome()
print('Generation', Generation)
print('Best Chromosome', Population[4])

li = [0]*100
li[Generation-1] = fitness_value(Population[4].x, Population[4].y)

while Generation < 100:
    parent = rouletteWheel(2)
    crossover(parent[0], parent[1])
    selection_survivor()
    BestChromosome()

    Generation += 1

    li[Generation-1] = fitness_value(Population[4].x, Population[4].y)
    print('Generation', Generation)
    print('Best Chromosome', Population[4])

# Graph labeling and representation
plt.plot(range(1, Generation + 1), li)
plt.xlim(left=0.0)
plt.ylim(bottom=0.0)
plt.title("Fitness Value Growth")
plt.ylabel("Fitness")
plt.xlabel("Generation")
plt.show()
```

4.0 - Output

```
Generation 1
Best Chromosome [0, 1, 0, 1, 0, 1] min(x,y): (2.142857142857143, -2.1428571428571432)  Heuristic value : 0.20801555615828518  Fitness value : 4.8073
32771011946
Generation 2
Best Chromosome [0, 1, 0, 1, 0, 1] min(x,y): (2.142857142857143, -2.1428571428571432)  Heuristic value : 0.20801555615828518  Fitness value : 4.8073
32771011946
Generation 3
Best Chromosome [0, 1, 0, 1, 0, 1] min(x,y): (2.142857142857143, -2.1428571428571432)  Heuristic value : 0.20801555615828518  Fitness value : 4.8073
32771011946
Generation 4
Best Chromosome [0, 1, 0, 1, 0, 1] min(x,y): (2.142857142857143, -2.1428571428571432)  Heuristic value : 0.20801555615828518  Fitness value : 4.8073
32771011946
Generation 5
Best Chromosome [0, 1, 0, 1, 0, 1] min(x,y): (2.142857142857143, -2.1428571428571432)  Heuristic value : 0.20801555615828518  Fitness value : 4.8073
32771011946
Generation 6
Best Chromosome [0, 1, 0, 1, 0, 1] min(x,y): (2.142857142857143, -2.1428571428571432)  Heuristic value : 0.20801555615828518  Fitness value : 4.8073
32771011946
Generation 7
Best Chromosome [0, 1, 0, 1, 0, 1] min(x,y): (2.142857142857143, -2.1428571428571432)  Heuristic value : 0.20801555615828518  Fitness value : 4.8073
32771011946
Generation 8
Best Chromosome [0, 1, 0, 1, 0, 1] min(x,y): (2.142857142857143, -2.1428571428571432)  Heuristic value : 0.20801555615828518  Fitness value : 4.8073
32771011946
Generation 9
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
1774673957336
Generation 10
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
1774673957336
Generation 11
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
1774673957336
Generation 12
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
1774673957336
Generation 13
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
1774673957336
Generation 14
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
1774673957336
Generation 15
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
```

```
Generation 15
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
1774673957336
Generation 16
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
1774673957336
Generation 17
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
1774673957336
Generation 18
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
1774673957336
Generation 19
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
1774673957336
Generation 20
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
1774673957336
Generation 21
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
1774673957336
Generation 22
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
1774673957336
Generation 23
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
1774673957336
Generation 24
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
1774673957336
Generation 25
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
1774673957336
Generation 26
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
1774673957336
Generation 27
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
1774673957336
Generation 28
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
1774673957336
Generation 29
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
```

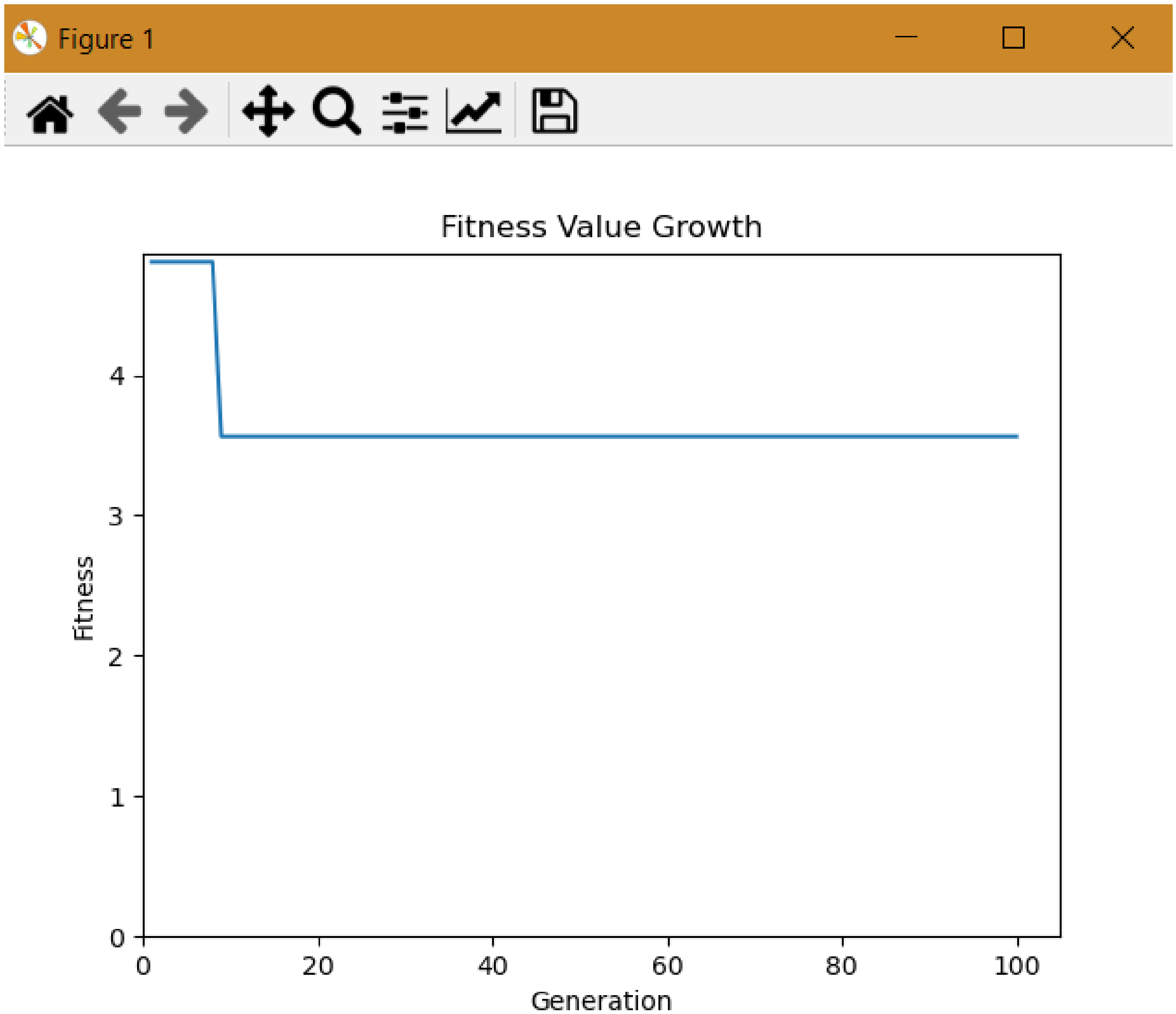
```
Generation 30
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
1774673957336
Generation 31
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
1774673957336
Generation 32
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
1774673957336
Generation 33
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
1774673957336
Generation 34
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
1774673957336
Generation 35
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
1774673957336
Generation 36
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
1774673957336
Generation 37
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
1774673957336
Generation 38
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
1774673957336
Generation 39
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
1774673957336
Generation 40
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
1774673957336
Generation 41
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
1774673957336
Generation 42
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
1774673957336
Generation 43
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
1774673957336
Generation 44
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
```

```
Generation 45
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
1774673957336
Generation 46
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
1774673957336
Generation 47
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
1774673957336
Generation 48
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
1774673957336
Generation 49
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
1774673957336
Generation 50
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
1774673957336
Generation 51
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
1774673957336
Generation 52
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
1774673957336
Generation 53
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
1774673957336
Generation 54
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
1774673957336
Generation 55
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
1774673957336
Generation 56
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
1774673957336
Generation 57
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
1774673957336
Generation 58
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
1774673957336
Generation 59
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
```

```
Generation 60
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
1774673957336
Generation 61
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
1774673957336
Generation 62
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
1774673957336
Generation 63
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
1774673957336
Generation 64
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
1774673957336
Generation 65
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
1774673957336
Generation 66
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
1774673957336
Generation 67
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
1774673957336
Generation 68
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
1774673957336
Generation 69
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
1774673957336
Generation 70
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
1774673957336
Generation 71
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
1774673957336
Generation 72
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
1774673957336
Generation 73
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
```

```
Generation 74
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
1774673957336
Generation 75
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
1774673957336
Generation 76
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
1774673957336
Generation 77
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
1774673957336
Generation 78
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
1774673957336
Generation 79
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
1774673957336
Generation 80
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
1774673957336
Generation 81
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
1774673957336
Generation 82
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
1774673957336
Generation 83
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
1774673957336
Generation 84
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
1774673957336
Generation 85
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
1774673957336
Generation 86
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
1774673957336
Generation 87
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
1774673957336
Generation 88
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
```

```
Generation 89
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
1774673957336
Generation 90
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
1774673957336
Generation 91
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
1774673957336
Generation 92
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
1774673957336
Generation 93
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
1774673957336
Generation 94
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
1774673957336
Generation 95
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
1774673957336
Generation 96
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
1774673957336
Generation 97
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564
1774673957336
Generation 98
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564177
4673957336
Generation 99
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564177
4673957336
Generation 100
Best Chromosome [1, 0, 1, 1, 0, 0] min(x,y): (-2.1428571428571432, -0.7142857142857144)  Heuristic value : 0.2805696431077766  Fitness value : 3.564177
4673957336
```



5.0 Instruction how to run the Readme.md

Click the link : https://github.com/nfasss/ArtificalIntelligence/blob/main/README.md

6.0 Presentation Video

Link for our presentation video : https://www.youtube.com/watch?v=U8YiT1Fqkhc

7.0 Task Distribution

Name	Task
MUHAMMAD FADLI RAMADHAN	- Presentation
NUR FASIHAH AYUNI BINTI MOHD YAHYA	- Report - Program Source Code - ReadMe.txt - Presentation