AI PROJECT REPORT

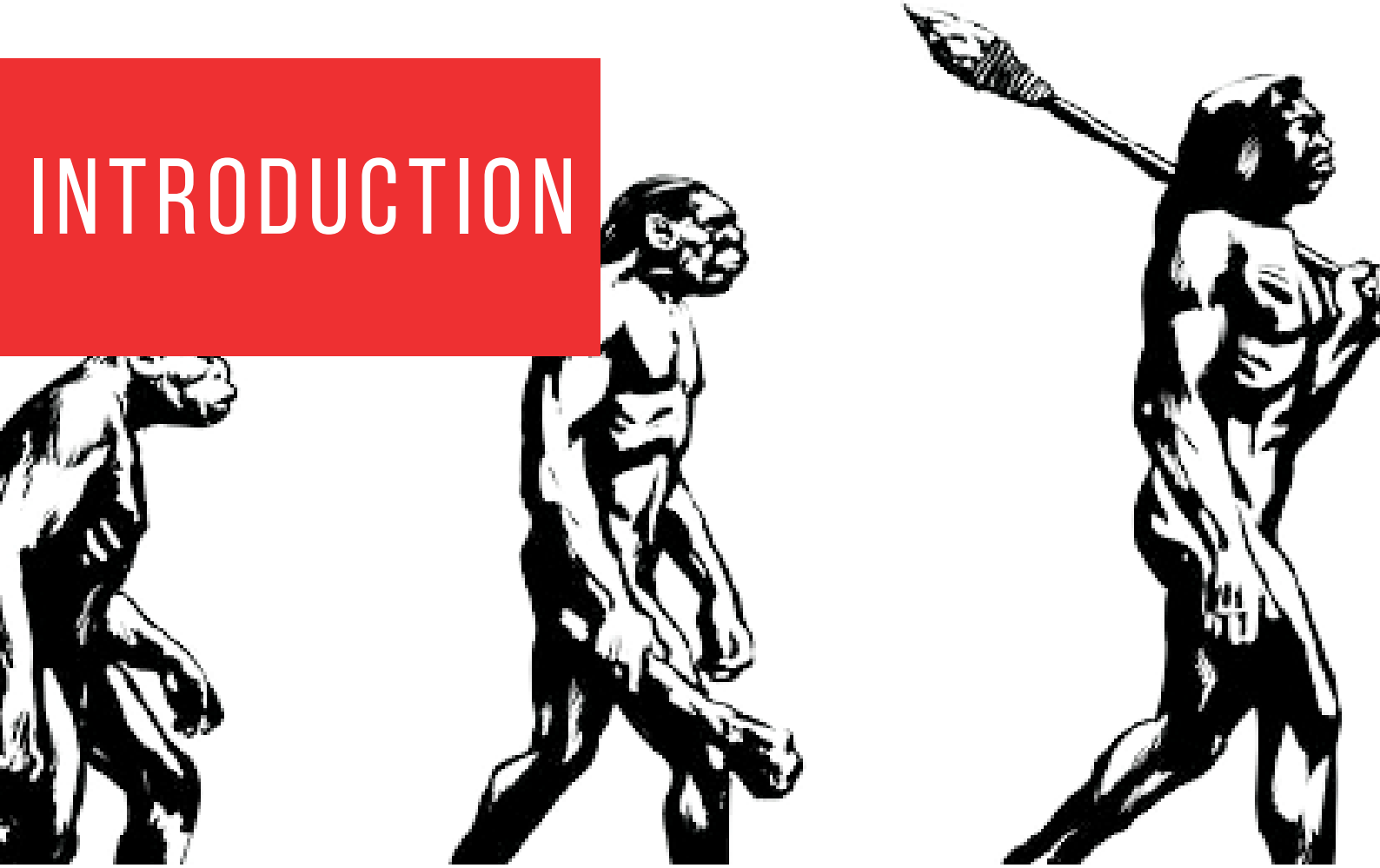
Evolutionary Programming

Guided by:

Ranjeet sir
CSE Dept
NIT Srinagar

Submited by:
(Group 5)
kushal siddharth
Janardhankarravula
Akshay Katti

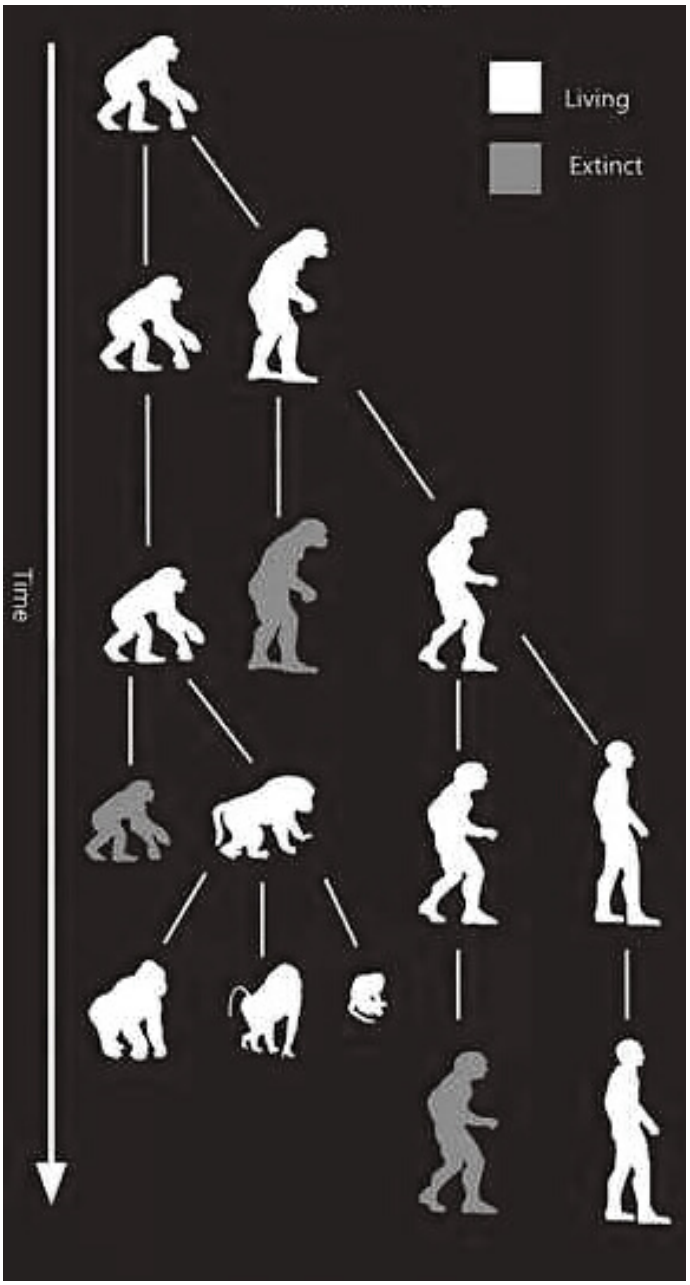
INTRODUCTION



The impact of evolutionary thinking on biology cannot be underestimated. But evolutionary thought extends beyond the study of life. Evolution is an optimization process that can be simulated on a computer and used for good engineering purpose. It also is essentially similar to the process of the scientific method, and as such it represents a procedure for generating machine intelligence. There are three main lines of investigation within the current framework of evolutionary computation:
(1) genetic algorithms,
(2) evolution strategies,
.

(3) evolutionary programming
Evolutionary programming resides in the latter category of simulation. Attention is placed on variation operations that can be constructed for a given representation so as to usefully adjust the behavior of the solutions Acc to data avaliable.
It is similar to genetic programming, but the structure of the program to be optimized is fixed, while its numerical parameters are allowed to evolve. It was first used by Lawrence J.

OVERVIEW



INITIALIZATION

In this stage we take N randomly genetared inputs with some characters like, Height, Weight, intelligence etc..

PARENT SELECTION

After the initialization of the test subjects. we do select the best N among them for the next generation to produce the Offsprings.

MUTATION

After the selection process, the test subjects who made the cut will produce the Off springs with certain Evolving factor for every physical & mental component.

SURVIUOR SELECTION

After the mutation stage, we need to evaluate the new offsprings and do a selection in round-robin fashion. And send the total 2N test subjects for the parent selection procces, Hence they Evolve.

APPLICATIONS

Evolutionary Computation has been applied to a wide range of real-life problems, ranging from telecommunication networks to complex systems, finance and economics, games, image analysis, evolutionary music, parameter optimization, bioinformatics, scheduling, and logistics.

1

GAMES

Natural evolution can be considered to be a game in which the rewards for an organism that plays a good game of life are the propagation of its genetic material to its successors and its continued survival. Expert game-playing strategies have been evolved without the need for human expertise.

2

IMAGE ANALYSIS

To research and review the techniques encompassed within image processing. To gain a thorough understanding of the principles involved in evolutionary computation and how they may be used. To compare evolutionary computation with competing image processing techniques, in the context of current research.

3

FINANCE AND ECONOMICS

The application of EC in finance pursues two main goals: first, to overcome the limitations of some theoretical models (and the strong assumptions being made by such models) and second, to innovate in this extremely competitive area of research.

PSUDO CODE

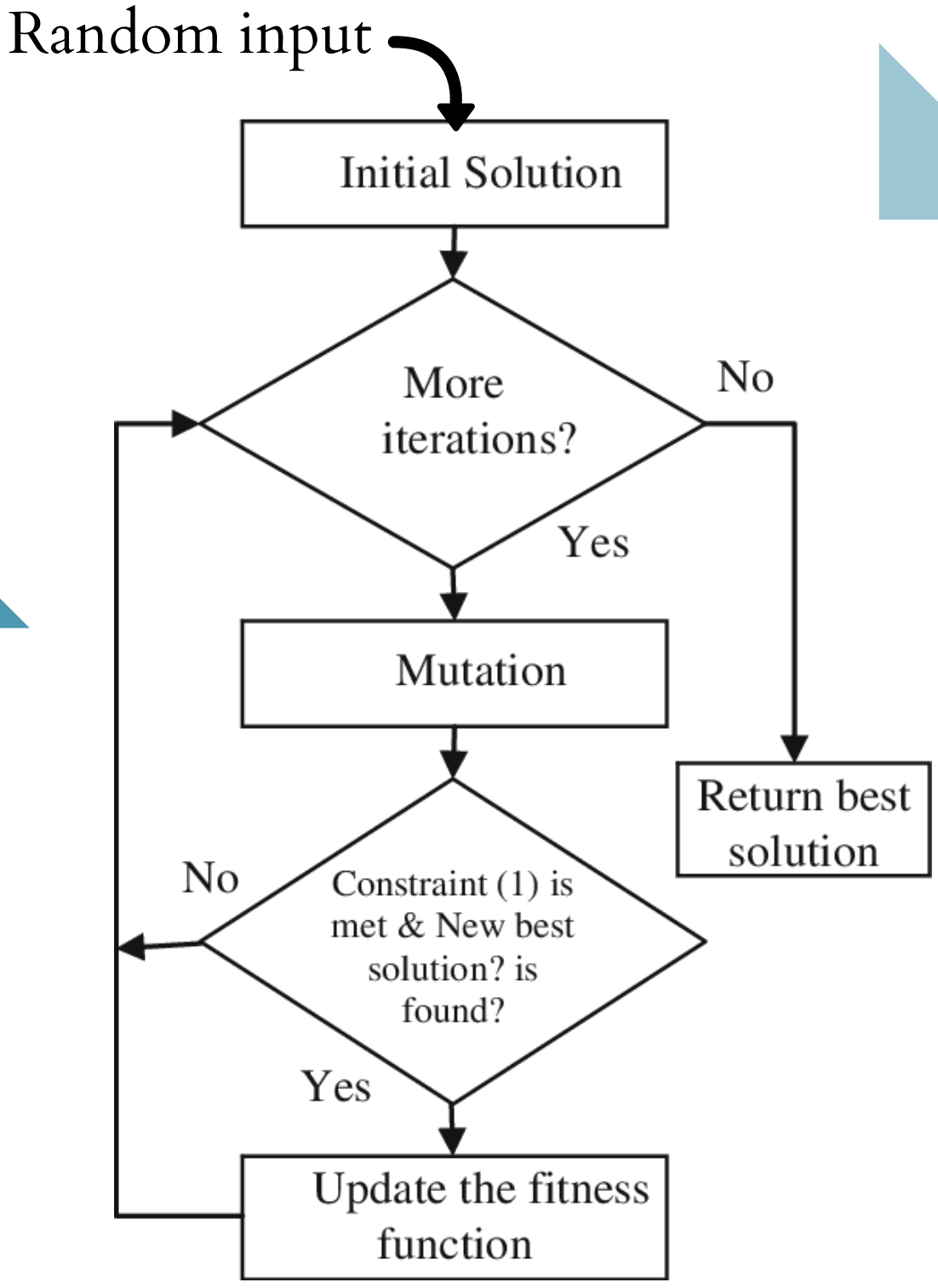
Step1: Start
Step2: Input: size of initial population.
Step3: Input:no.of iterations.
Step4: Initialize chromosomes to list of chromosomes
Step5:Craete Class Chromosome:
Properties of chromosome:
*height
*weight
*Intelligence
*mutation factor
*bmi
*fitness
*intel_score
*bmi_score
Step6: Function init_chromosomes(count=size of population):
{
for i in range(count):
Append chromosome to list of chromosomes
}

Step7: Create a function Function mutate() which mutates the properties of chromosome.
*Calculate mutation factor.
*Calculate muatated height {Ht*mf*2 (or) Ht/(mf*2)}
*Calculate mutated weight{wt*mf*2 (or)wt/(mf*2)}
*Calculate mutated intelligence(intel*mf*2 (or) intel/(mf*2))
*Add mutated chromosomes to existing chromosomes.

Step8:Create a Fuction Calculate finess() which calculates fitness values of each chromosome.
*Calculate intelligence score
{(intelscore/maximum_intelligence)/100}
*Calculate bmiscore{100-(meanBmi/maxbmi)*100}
*Calculate fitness value(intelscore+bmiscore)
Step9: Select the best among chromosomes according to fitness values .
Step10: If no. of iterations done were less than no. of iterations we given as input GO TO STEP 6.

Step11:Stop.

FLOW CHART

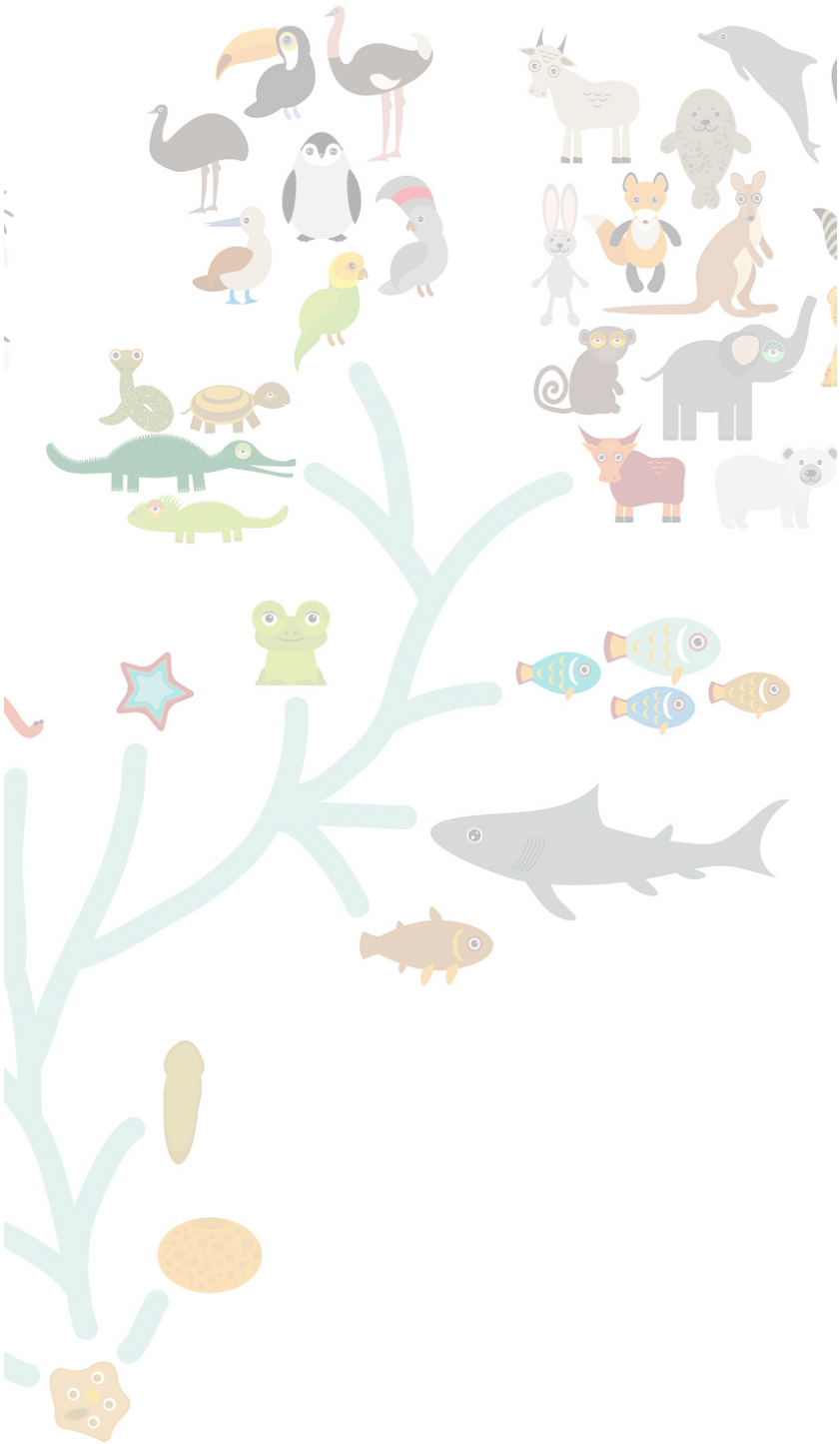

Random input
Initial Solution
More iterations?
No
Yes
Mutation
Return best solution
Constraint (1) is met & New best solution? is found?
No
Yes
Update the fitness function

1st Iteration for user input

Step 1: Initialise chromosome with randomly generated population

*** bmi = weight (kg) / height^2 (m^2)**
*** meanBmi = |bmi - 21.7|**
chromosomes = [
p1[ht=5, wt=50, intel=100, bmi=22.2, meanBmi=0.25, intel_score=0, bmi_score=0, fitness=0, mf=0.6]
p2[ht=3, wt=40, intel=250, bmi=49.3, meanBmi=27.6, intel_score=0, bmi_score=0, fitness=0, mf=0.45]
p3[ht=8, wt=90, intel=10, bmi=22.2, meanBmi=6.06, intel_score=0, bmi_score=0, fitness=0, mf=0.5]
]

Iteration 1:



child 1:
c1.ht = p1.ht * 0.6 * 2 = 6.0
c1.wt = p1.wt * o.6 * 2 = 60
c1.intel = p1.intel * 0.6 * 2 = 120
c1.bmi = 18.5

child 2:
c2.ht = p2.ht * 0.45 * 2 = 2.7
c2.wt = p2.wt * 0.45 * 2 = 36
c2.intel = p2.intel * 0.45 * 2 = 225
c2.bmi = 54.8

child 3:
c3.ht = p3.ht * 0.5 * 2 = 8
c3.wt = p3.wt * 0.5 * 2 = 90
c3.intel = p3.intel * 0.5 * 2 = 10
c3.bmi = 15.62

p1.intel_score = 100 / 250 * 100 = 40
p2.intel_score = 250 / 250 * 100 = 100
p3.intel_score = 10 / 250 * 100 = 4
p4.intel_score = 120 / 250 * 100 = 48
p5.intel_score = 225 / 250 * 100 = 90
p6.intel_score = 10 / 250 * 100 = 4

p1.bmi_score = 100 - (0.25 / 33.1 * 100) = 99.24
p2.bmi_score = 100 - (27.6 / 33.1 * 100) = 16.61
p3.bmi_score = 100 - (6.06 / 33.1 * 100) = 81.69
p4.bmi_score = 100 - (27.6 / 33.1 * 100) = 90.33
p5.bmi_score = 100 - (33.1 / 33.1 * 100) = 0
p6.bmi_score = 100 - (6.06 / 33.1 * 100) = 81.69

chromosomes = {

p1[ht=5, wt=50, intel=100, bmi=22.2, meanBmi=0.25, intel_score=40, bmi_score=99.24, fitness=139.24, mf=0.6]

p2[ht=3, wt=40, intel=250, bmi=49.3, meanBmi=27.6, intel_score=100, bmi_score=16.61, fitness=116.61, mf=0.45]

p3[ht=8, wt=90, intel=10, bmi=22.2, meanBmi=6.06, intel_score=4, bmi_score=81.69, fitness=85.69, mf=0.5]

p4[ht=6, wt=60, intel=120, bmi=18.5, meanBmi=3.2, intel_score=48, bmi_score=90.33, fitness=138.33, mf=0.55]

p5[ht=2.7, wt=36, intel=225, bmi=54.8, meanBmi=33.1, intel_score=90, bmi_score=0, fitness=90, mf=0.65]

p6[ht=8, wt=90, intel=10, bmi=15.62, meanBmi=6.06, intel_score=4, bmi_score=81.69, fitness=85.69, mf=0.45]

}

#selection

chromosomes = [

p1[ht=5, wt=50, intel=100, bmi=22.2, meanBmi=0.25, intel_score=40, bmi_score=99.24, fitness=139.24, mf=0.6]

p4[ht=6, wt=60, intel=120, bmi=18.5, meanBmi=3.2, intel_score=48, bmi_score=90.33, fitness=138.33, mf=0.55]

p2[ht=3, wt=40, intel=250, bmi=49.3, meanBmi=27.6, intel_score=100, bmi_score=16.61, fitness=116.61, mf=0.45]

]

CODE

main.py repo.py

```python
from random import randint

MEAN_BMI = 21.7

max_mean = 0.0
max_intelligence = 0.0
chromosomes = []
gen = 1


def bmi(height, weight):
    bmi_value = float(weight) / ((height * 0.3) ** 2)

    if bmi_value >= 1:
        return bmi_value
    else:
        return 1.0


def get_mutation_factor():
    return randint(30, 100) / 100.0


class Chromosome:
    def __init__(self, **kwargs):
        global max_mean
        global max_intelligence
        if len(kwargs) == 0:
            self.height = randint(1, 10) + 1.0 / randint(1, 100)  # height in feet
            self.weight = randint(1, 300)  # weight in kg
            self.intelligence = randint(0, 100)
        else:
            self.height = kwargs['height']
            self.weight = kwargs['weight']
            self.intelligence = kwargs['intelligence']

        self.mutation_factor = get_mutation_factor()
        self.Bmi = bmi(self.height, self.weight)
        self.meanBmi = abs(self.Bmi - MEAN_BMI)
        self.fitness = 0
        self.intel_score = 0
        self.bmi_score = 0

        if self.meanBmi > max_mean:
            max_mean = self.meanBmi
        if self.intelligence > max_intelligence:
            max_intelligence = self.intelligence


def get_meanBmi(person):
    return person.meanBmi

def get_fitness(person):
    return person.fitness


def init_chromosomes(count):
    for i in range(count):
        chromosomes.append(Chromosome())


def mutate_height(height, mf):
    prob = randint(0, 1)

    if prob:
        ht = height * mf * 2
        if ht > 10.0:
            return 10.0
        else:
            return ht
    else:
        try:
            return height / (mf * 2)
        except ZeroDivisionError:
            return height
```

CODE

```python
def mutate_weight(weight, mf):
    prob = randint(0, 1)

    if prob:
        return weight * mf * 2
    else:
        try:
            return weight / (mf * 2)
        except ZeroDivisionError:
            return weight


def mutate_intelligence(intelligence, mf):
    prob = randint(0, 1)

    if prob:
        intel = intelligence * mf * 2

        if intel > 300.0:
            return 300.0
        else:
            return intel
    else:
        try:
            if mf >= 0.5:
                return intelligence / (mf * 2)
            else:
                return intelligence
        except ZeroDivisionError:
            return intelligence


def mutate():
    global chromosomes
    mutated_chromosomes = []
    for chromo in chromosomes:
        height = mutate_height(chromo.height, chromo.mutation_factor)
        weight = mutate_weight(chromo.weight, chromo.mutation_factor)
        intelligence = mutate_intelligence(chromo.intelligence, chromo.mutation_factor)
        new_chromosome = Chromosome(height=height, weight=weight, intelligence=intelligence)
        mutated_chromosomes.append(new_chromosome)

    chromosomes.extend(mutated_chromosomes)


def cal_fitness():
    for chromo in chromosomes:
        chromo.intel_score = chromo.intelligence / max_intelligence * 100
        chromo.bmi_score = 100.0 - chromo.meanBmi / max_mean * 100

        chromo.fitness = chromo.intel_score + chromo.bmi_score


def print_gen():
    global max_bmi
    current_gen = []
    print(f'Gen {gen}')
    for chromo in chromosomes:
        print(f'intel_score={round(chromo.intel_score, 2)}  bmi_score={round(chromo.bmi_score, 2)}'
              f' intel={round(chromo.intelligence, 2)} bmi={round(chromo.Bmi, 2)}')

    print(current_gen)


# ----------------------------------------------------------------------------------------------------

# initialize chromosomes
n = int(input('Enter the size of the initial population size : '))
itr = int(input('Enter no of iteration : '))

init_chromosomes(n)
print_gen()


# evolve

def evolve():
    global chromosomes
    global gen
```

CODE

```
    gen += 1

    # mutate chromosomes
    mutate()

    # fitness calculation
    cal_fitness()

    # survivor selection
    chromosomes = sorted(chromosomes, key=get_fitness, reverse=True)
    chromosomes = chromosomes[0:n]

    # display current generation data
    print_gen()


    if gen < itr:
        evolve()


evolve()
```

GIT LINK

RESULT

```
main
/Users/janardhankarravula/PycharmProjects/EP/venv/bin/python /Users/jar
Enter the size of the initial population size : 10
Enter no of iteration : 10
Gen 1
intel_score=0  bmi_score=0 intel=0 bmi=5.88
intel_score=0  bmi_score=0 intel=14 bmi=138.89
intel_score=0  bmi_score=0 intel=56 bmi=6.29
intel_score=0  bmi_score=0 intel=72 bmi=66.64
intel_score=0  bmi_score=0 intel=7 bmi=258.77
intel_score=0  bmi_score=0 intel=77 bmi=11.42
intel_score=0  bmi_score=0 intel=68 bmi=116.29
intel_score=0  bmi_score=0 intel=92 bmi=54.49
intel_score=0  bmi_score=0 intel=65 bmi=82.3
intel_score=0  bmi_score=0 intel=52 bmi=105.88
```

```
Gen 2
intel_score=65.85  bmi_score=93.71 intel=99.36 bmi=48.29
intel_score=60.98  bmi_score=92.24 intel=92 bmi=54.49
intel_score=51.03  bmi_score=97.57 intel=77 bmi=11.42
intel_score=100.0  bmi_score=48.25 intel=150.88 bmi=240.37
intel_score=47.97  bmi_score=97.11 intel=72.38 bmi=9.49
intel_score=47.72  bmi_score=89.36 intel=72 bmi=66.64
intel_score=37.12  bmi_score=96.35 intel=56 bmi=6.29
intel_score=32.15  bmi_score=97.04 intel=48.51 bmi=34.21
intel_score=43.08  bmi_score=85.66 intel=65 bmi=82.3
intel_score=28.89  bmi_score=97.89 intel=43.59 bmi=30.63
```

```
Gen 10
intel_score=100.0  bmi_score=99.69 intel=300.0 bmi=20.39
intel_score=100.0  bmi_score=99.1 intel=300.0 bmi=25.49
intel_score=100.0  bmi_score=99.1 intel=300.0 bmi=25.49
intel_score=100.0  bmi_score=99.08 intel=300.0 bmi=17.83
intel_score=98.57  bmi_score=98.78 intel=295.72 bmi=16.55
intel_score=98.0  bmi_score=99.22 intel=294.0 bmi=24.98
intel_score=100.0  bmi_score=96.26 intel=300.0 bmi=5.9
intel_score=95.2  bmi_score=98.24 intel=285.59 bmi=14.25
intel_score=95.73  bmi_score=97.61 intel=287.18 bmi=11.58
intel_score=95.2  bmi_score=95.83 intel=285.59 bmi=4.1
[]

Process finished with exit code 0
```

Thank you

AI PROJECT REPORT:

Kushal Siddharth
Janardhan karravula
Akshay katti