A.Y. 2022-2023

Subject: Artificial Intelligence **SAP ID: 60004220253 – Devansh Mehta**

Experiment No 05

Aim: Implementation of Genetic Algorithm

Theory:

Genetic algorithms are optimization algorithms inspired by the process of naturalselection. They involve evolving a population of potential solutions to a problemover multiple generations. Here's a brief overview:

1. **Initialization**: Start with a population of potential solutions (chromosomes) randomly generated.
2. **Selection**: Evaluate the fitness of each solution and select individuals to serve as parents based on their fitness. Solutions with higher fitness have ahigher chance of being selected.
3. **Crossover (Recombination)**: Pair selected parents and create offspring bycombining their genetic information. This mimics the crossover of geneticmaterial in natural reproduction.
4. **Mutation**: Introduce small random changes to the offspring's genetic information to promote diversity, similar to genetic mutations in nature.
5. **Evaluation**: Assess the fitness of the new population.
6. **Replacement**: Create a new population for the next generation by selectingindividuals from the current population and the offspring.
7. **Termination**: Repeat the process for a predefined number of generations or until a satisfactory solution is found.

Code:

```
from random import randint


def selection(li):
  dec = list(map(lambda x : int(x, 2), li))


  fit = list(map(lambda x : x*x, dec))
```

```
  s = sum(fit)
  prob = list(map(lambda x : round(x/s, 3), fit))

  avg = s/n
  exe = list(map(lambda x : round(x/avg, 3), fit))

  ac = list(map(lambda x : round(x), exe))

  return dec, fit, prob, exe, ac

def pp(li, ac, n):
  co = [] temp
  = [] index =
  []
  for i in range(n):
    if ac[i] == 1:
    co.append(li[i])
    elif ac[i] >= 2:
      for j in range(ac[i] - 1):
      temp.append(li[i])
      co.append(li[i])
    elif ac[i] == 0 and len(temp) != 0:
      co.append(temp[0]) temp.pop(0)
    elif ac[i] == 0 and len(temp) == 0:
      index.append(i)
  if len(index) != 0 and len(temp) != 0:
```

```
    for i in index:
      co.insert(i, temp[0])
      temp.pop(0)
  elif len(index) != 0 and len(temp) == 0:
    co.insert(i, li[i])
  return co

def cr(x):
  s = 0
  for i in x:
    if i == '1':
      s = s + 1
  return s

def crossing(li, n):
  crossed = []
  for i in range(0, n, 2):
    temp1 = li[i]j
    = i + 1 temp2
    = li[j]
    crosspoint = cr(temp1)
    print("The crosspoint for pair " + str(i) + " is " + str(crosspoint))temp3 =
    temp1[crosspoint: ]
    temp4 = temp2[crosspoint: ]
    temp1 = temp1[0 : crosspoint] + temp4temp2
    = temp2[0 : crosspoint] + temp3
    crossed.append(temp1)
```

```
    crossed.append(temp2)
  return crossed

def mutation(li, n):
  mut = [] for
  i in li:
    j = randint(0, n - 1)
    print("For pair " + str(i) + ", the bit that will be changed is " + str(j))if i[j] == '1':
      i = i[0 : j] + '0' +i[j + 1 : ]
      mut.append(i)
    elif i[j] == '0':
      i = i[0 : j] + '1' +i[j + 1 : ]
      mut.append(i)
  return mut

n = int(input("Enter number of samples: "))sam =
[]
for i in range(n): sam.append(input("Enter
  gene: "))

m = int(input("Enter number of generations to be computed: "))crossed =
sam.copy()
for i in range(m):
  dec, fit, prob, exe, ac = selection(crossed)s =
  sum(ac)
  if s < n:
```

```
    maxi = max(ac)
    k = ac.index(maxi - 1)
    ac[k] += 1
  if s > n:
    maxi = max(ac)
    k = ac.index(maxi)
    ac[k] -= 1
  print("\n_____GENERATION ", i, "_____
_____")
  print("Initial Population\tX Value\t\tFitness Value\tProbability\tExpected
Count\t\tActual Count")
  for j in range(n):
    print(crossed[j], "\t\t", dec[j], "\t\t", fit[j], "\t", prob[j], "\t\t", exe[j], "\t\t\t",ac[j])
  co = pp(crossed, ac, n)
  print("\nSelected Genes for Crossover - \n", co)crossed
  = crossing(co, n)
  print("\nCrossover - \n", crossed)
  crossed = mutation(crossed, n)
  print("\nMutated - \n", crossed)
print("\nGENERATION ", (m + 1), " - ", crossed)
```

A.Y. 2022-2023

Subject: Artificial Intelligence **SAP ID: 60004220253 – Devansh Mehta**

```
PS C:\Users\devan\OneDrive\Desktop\AI Prac Codes> python -u "c:\Users\devan\Downloads\Hill (1).py"
Enter number of samples: 4
Enter gene: 1010
Enter gene: 1100
Enter gene: 1111
Enter gene: 0000
Enter number of generations to be computed: 2

------------------------------------------------ GENERATION  0 ------------------------------------------------
Initial Population      X Value         Fitness Value   Probability     Expected Count          Actual Count
1010            10              100     0.213           0.853                   1
1100            12              144     0.307           1.228                   1
1111            15              225     0.48            1.919                   2
0000            0               0       0.0             0.0                     0

Selected Genes for Crossover -
 ['1010', '1100', '1111', '1111']
The crosspoint for pair 0 is 2
The crosspoint for pair 2 is 4

Crossover -
 ['1000', '1110', '1111', '1111']
For pair 1000, the bit that will be changed is 0
For pair 1110, the bit that will be changed is 0
For pair 1111, the bit that will be changed is 0
For pair 1111, the bit that will be changed is 1

Mutated -
 ['0000', '0110', '0111', '1011']
```

```
------------------------------------------------ GENERATION  1 ------------------------------------------------
Initial Population      X Value         Fitness Value   Probability     Expected Count          Actual Cou
0000            0               0       0.0             0.0                     0
0110            6               36      0.175           0.699                   1
0111            7               49      0.238           0.951                   1
1011            11              121     0.587           2.35                    2

Selected Genes for Crossover -
 ['1011', '0110', '0111', '1011']
The crosspoint for pair 0 is 3
The crosspoint for pair 2 is 3

Crossover -
 ['1010', '0111', '0111', '1011']
For pair 1010, the bit that will be changed is 2
For pair 0111, the bit that will be changed is 3
For pair 0111, the bit that will be changed is 1
For pair 1011, the bit that will be changed is 3

Mutated -
 ['1000', '0110', '0011', '1010']

GENERATION  3  -  ['1000', '0110', '0011', '1010']
```

Conclusion: In conclusion, Genetic algorithms efficiently explore solution spaces, demonstrating versatility and suitability for complex problems. However,their computational intensity, sensitivity to parameter choices, and the absence ofoptimality guarantees should be considered in their application.