CS571 - ARTIFICIAL INTELLIGENCE LAB

Lab - 5
Genetic Algorithm v/s Simulated Annealing

Jenish Monpara 1901CS28
Tanishq Malu 1901CS63
Tarusi Mittal 1901CS65

Instructions to run the code

1. Launch terminal
2. Run “python3 main.py Start_State.txt End_State.txt”

Sample Input-Output for each case for Success and failure cases:

1) Initial Population (assume to be 10)
2) Selection (use Roulette Wheel Selection)
3) Crossover (high probability value to be chosen, usually above 0.6)
4) Mutation (low probability value to be chosen, usually below 0.2)
5) Fitness function: Devise your own two fitness functions.

Start State:
P1: [5,9,8,4,2,1,7,3,6]
P2: [B,D, E, A, F,C,G,H]

Goal State:
[1,2,3,4,5,6,7,8,B]

Que 2. Compare and contrast (with justification) the results obtained from the two different Fitness functions. Justify the fitness function choice. Why was it chosen and how is it handling the problem in hand?

Ans:

```
45 # Fitness function 2
46 def find_fitness2(population):
47     length = len(population)
48     fitness = []
49     for i in range(length):
50         score = 0
51         for j in range(9):
52             if population[i][j] == goal[j]:
53                 score += 20
54             elif population[i][j] in goal:
55                 score += 10
56             elif population[i][j] == '_':
57                 score -= 300
58         fitness.append([score,i])
59     return fitness
60
```

Fitness function 2 is a quite generic fitness function which simply reward whenever a good gene is incorporated(good gene is the one which is required in the solution).

```
25 def find_fitness1(population):
26     length = len(population)
27     fitness = []
28     for i in range(length):
29         score = 0
30         for j in range(9):
31             if population[i][j] == goal[j]:
32                 score += 20
33             elif population[i][j] in goal:
34                 score += 10
35             elif population[i][j] == '_':
36                 score -= 300
37             else:
38                 score -= 30
39         for j in range(9):
40             if population[i].count(population[i][j])>1:
41                 score -= 10
42         fitness.append([score,i])
43     return fitness
```

Fitness function 1 is a more enhanced and strict fitness function which penalizes wrong gene selection and even duplicate gene selection which one may not require in an organism.(Having different genes for better functionality is better than having similar genes incorporating redundancy).

This helps in finding the solution at a much faster rate and in less number of generations. Also, on multiple runs the frequency of attaining the output has increased. However, as we know about the problem of overfitting in ML, the fitness function tends to overfit for the specific kind of solution we are aiming for however survival of the fittest itself sets some parameters which needs to be passed on to the generation and hence some amount of overfit is a natural outcome in such problems.

Que 3. Compare your results obtained in the Simulated Annealing implementations (from previous assignment) with the Genetic Algorithm implementations.
a. Take multiple examples (at least 3) of the same start state and goal state combinations and compare both algorithms.
Ans: Taking same start and end goal:

EXAMPLE 1:

Parent: [5,9,8,4,2,1,7,3,6]
Goal : [1,2,3,4,5,6,7,8,9]

1. **Simulated Annealing :**

```
tanishqmalu@TanishqMalu:/mnt/d/tanishq/4th year/ai lab/lab4$ python3 main.py comp_start_1.txt comp_end.txt
Enter the type of Hueristic function:
1. h1(n) = Number of tiles displaced from their destined position.
2. h2(n) = Sum of Manhattan distance of each tile from the goal

1
Enter the max temperature: 1000
1. Exponential decay
2. Linear decay
3. Logarithmic decay
3
OOPS ! The program was unable to find a solution !


Start State of the Puzzle:
T5 B T8
T4 T2 T1
T7 T3 T6


Goal State of the Puzzle:
T1 T2 T3
T4 T5 T6
T7 T8 B


Total number of states explored before ending the program: 49600
tanishqmalu@TanishqMalu:/mnt/d/tanishq/4th year/ai lab/lab4$
```

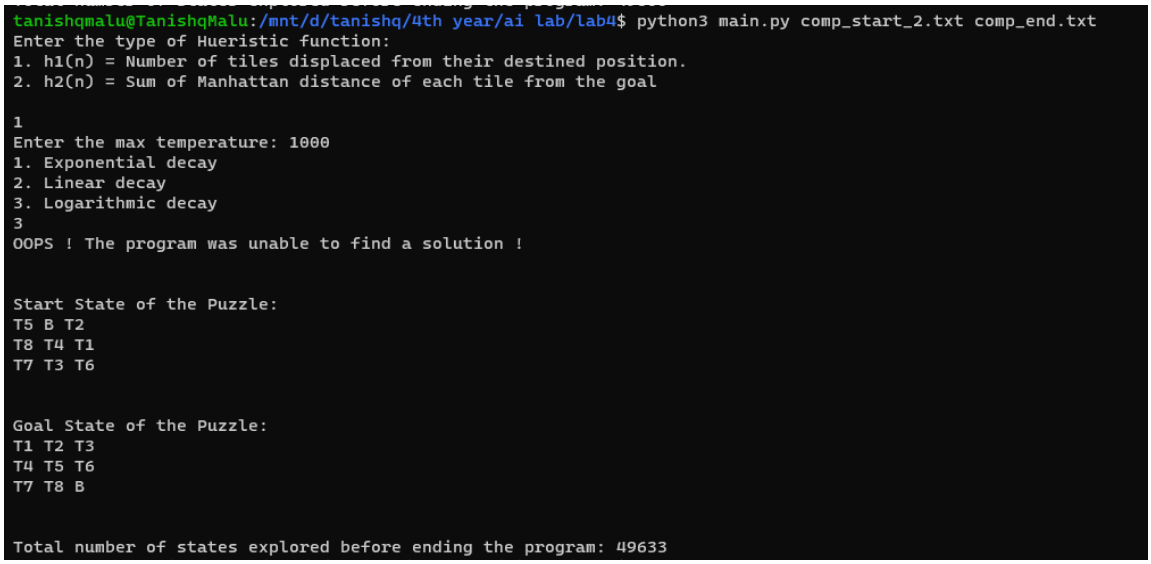
2. **Genetic Algorithm:**

```
tanishqmalu@TanishqMalu:/mnt/d/tanishq/4th year/ai lab/lab5$ python3 main.py
Solution found
Start state:
Adam:  ['5', '9', '8', '4', '2', '1', '7', '3', '6']
Eve:  ['5', '9', '8', '4', '2', '1', '7', '3', '6']
End State:  ['1', '2', '3', '4', '5', '6', '7', '8', 'B']
Number of States explored: 8149
Number of steps required: 815
Time taken: 0.3680593967437744
tanishqmalu@TanishqMalu:/mnt/d/tanishq/4th year/ai lab/lab5$
```

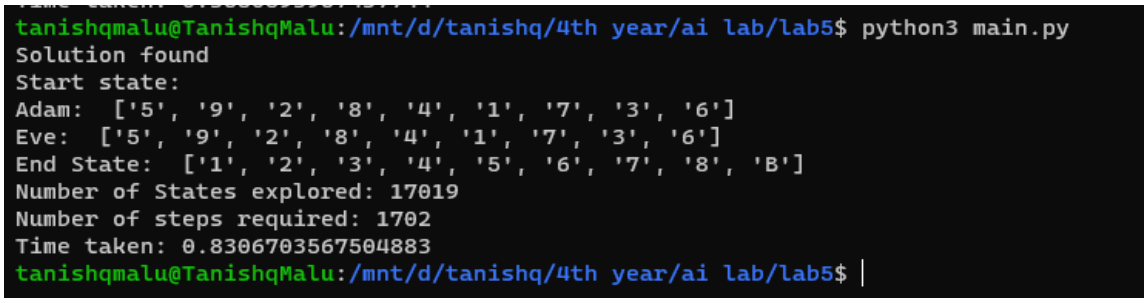
EXAMPLE 2:

Parent: [5,9,2,8,4,1,7,3,6]
Goal : [1,2,3,4,5,6,7,8,9]

1. **Simulated Annealing:**



2. **Genetic Algorithm:**



EXAMPLE 3:

Parent: [9,2,3,1,4,5,6,7,8]
Goal : [1,2,3,4,5,6,7,8,9]

1. Simulated Annealing:

```
tanishqmalu@TanishqMalu:/mnt/d/tanishq/4th year/ai lab/lab4$ python3 main.py comp_start_3.txt End_State.txt
Enter the type of Hueristic function:
1. h1(n) = Number of tiles displaced from their destined position.
2. h2(n) = Sum of Manhattan distance of each tile from the goal

1
Enter the max temperature: 1000
1. Exponential decay
2. Linear decay
3. Logarithmic decay
1
Found a solution to the puzzle !


Start State of the Puzzle:
B T2 T3
T1 T4 T5
T6 T7 T8


Goal State of the Puzzle:
T2 T4 T3
T1 B T5
T6 T7 T8


Total number of states explored: 3
Total number of states to Optimal Path: 3
Optimal Path Cost: 2
Time taken: 0.0012753009796142578

B T2 T3
T1 T4 T5
T6 T7 T8

   V

T2 B T3
T1 T4 T5
T6 T7 T8

   V

T2 T4 T3
T1 B T5
T6 T7 T8
tanishqmalu@TanishqMalu:/mnt/d/tanishq/4th year/ai lab/lab4$
```

2. Genetic Algorithm:

```
tanishqmalu@TanishqMalu:/mnt/d/tanishq/4th year/ai lab/lab5$ python3 main.py
Solution found
Start state:
Adam:  ['9', '2', '3', '1', '4', '5', '6', '7', '8']
Eve:  ['9', '2', '3', '1', '4', '5', '6', '7', '8']
End State:  ['2', '4', '3', '1', 'B', '5', '6', '7', '8']
Number of States explored: 15379
Number of steps required: 1538
Time taken: 0.8902201652526855
tanishqmalu@TanishqMalu:/mnt/d/tanishq/4th year/ai lab/lab5$ python3 main.py
Solution found
Start state:
Adam:  ['9', '2', '3', '1', '4', '5', '6', '7', '8']
Eve:  ['9', '2', '3', '1', '4', '5', '6', '7', '8']
End State:  ['2', '4', '3', '1', 'B', '5', '6', '7', '8']
Number of States explored: 13889
Number of steps required: 1389
Time taken: 1.3274102210998535
tanishqmalu@TanishqMalu:/mnt/d/tanishq/4th year/ai lab/lab5$ python3 main.py
No Solution found in 2000 generations.
Start state:
Adam:  ['9', '2', '3', '1', '4', '5', '6', '7', '8']
Eve:  ['9', '2', '3', '1', '4', '5', '6', '7', '8']
End State:  ['2', '4', '3', '1', 'B', '5', '6', '7', '8']
Number of States explored: 19989
Time taken: 0.8782615661621094
```

b. Analyze the results obtained with proper justifications.

Ans: As we can see in above examples, the genetic algorithm produces quite random results and is dependent upon factors like mutation and crossover rate along with its high dependency on fitness criteria. A state may or may not be reachable and the answer may vary on different iterations for genetic algorithms with higher randomness than simulated annealing. Simulated annealing provided significantly consistent results and gives us more surety.

c. Describe your results on both algorithms and state the reasons for the difference of approach in both algorithms.

Ans: Simulated annealing uses a probabilistic approach for selection of next neighbors while giving a chance to even downhill moves. However this probability decreases as the temperature cools down. In genetic algorithms the probability of mutation and crossover rate remains the same in basic implementation of genetic algorithms. (There are advanced implementations which change mutation and crossover rate too, but that's out of scope of the current problem).

Genetic algorithm provides randomness on results and gives both positive and negative solutions. Simulated annealing seems more consistent in this case and gives more trustful results.

Genetic algorithm's result may vary depending upon the number of iteration/generation the algo is run for. Simulated annealing is independent of this parameter.

Fitness factor and cool down factor are quite analogous in the two algos and they quite prominently drive the problem towards the solution. Good fitness factors helps select better offspring and better cooldown factor increases chances of selecting bad/downhill moves too.

d. Describe your views on what algorithm should have performed better for this particular problem and does your intuition match the results?

Ans: We personally thought that simulated annealing would have worked better in terms of consistency and genetic algorithms would be able to find solutions to the problems. The intuition was based on the fact that mutation in genetic

algorithms may help teleport from a dead end state to state which can lead to a solution. Our intuition was quite in line with the results we got.

e. Is it infeasible to use Genetic Algorithm for an 8-Puzzle problem? If yes, Justify

Ans: The way Genetic algorithm generates the offspring or neighbors in search space involves crossover and mutation which does not follow the restriction of 8-puzzle problem, i.e. the restriction of one adjacent swap with blank tile is violated and hence it is infeasible to use genetic algorithm for 8-puzzle problem.

END