COMSATS University, Islamabad

Class: RAI
Subject: AIC530 – Programming for AI
Instructor: Dr. Muhammad Imran

Date: 26-03-24

Genetic Algorithms

In this lab, students have to implement a type of local search algorithm known as Genetic Algorithm to solve the eight-queens problem.

As discussed in the lecture, genetic algorithm is a type of local search algorithm which does not keeps track of the whole search path rather it moves from solution to solution in the space of candidate solutions (the *search space*) by applying local changes, until a solution deemed optimal is found or a time bound is elapsed. As a result, these types of algorithms have two main advantages:

- They use very little memory—usually a constant amount; and
- They can often find reasonable solutions in large or infinite (continuous) state spaces for which systematic search algorithms are unsuitable.

To be specific, there are four iterative steps involved in implementation of genetic algorithm other than the ***Initialization*** at the start of the algorithm.

1. Fitness Evaluation
2. Selection
3. Crossover
4. Mutation

To implement the genetic algorithms in the context of eight-queens problem, our target fitness function is to arrange eight queens on the **8*8** board such that none of the queen pairs attack each other on the board. The potential ***Initialization*** in Python may be to first declare eight nested lists, within a main list, with eight columns having zeros at all places in it. Then randomly replacing one of the zeros in each column with the value of "1" to depict the placement of the queen in that column. Remember, there should be only one queen in each column, so as to minimize the number of possible attacks due to queens being in the same column. One such sample is shown below, while students can generate their own one if they feel they have a got better alternative solution.

```
Board: 1
```

```
[[0 0 0 1 0 0 0 0]
 [0 1 0 0 0 0 0 0]
 [0 0 1 0 0 0 0 0]
 [0 0 0 0 0 1 0 0]
 [0 0 1 0 0 0 0 0]
 [0 0 0 0 0 0 1 0]
 [0 0 0 0 1 0 0 0]
 [0 1 0 0 0 0 0 0]]
```

```
Board: 2
```

```
[[0 0 0 1 0 0 0 0]
 [0 0 0 0 0 0 0 1]
 [0 0 1 0 0 0 0 0]
```

Genetic Algorithms

```
 [0 1 0 0 0 0 0 0]
 [0 1 0 0 0 0 0 0]
 [0 1 0 0 0 0 0 0]
 [0 0 0 1 0 0 0 0]
 [1 0 0 0 0 0 0 0]]

Board: 3

[[0 0 1 0 0 0 0 0]
 [0 0 0 0 0 1 0 0]
 [0 0 1 0 0 0 0 0]
 [0 1 0 0 0 0 0 0]
 [0 0 0 0 0 1 0 0]
 [0 1 0 0 0 0 0 0]
 [0 1 0 0 0 0 0 0]
 [0 1 0 0 0 0 0 0]]

Board: 4

[[0 0 0 0 0 1 0 0]
 [0 0 0 0 0 0 0 1]
 [1 0 0 0 0 0 0 0]
 [0 1 0 0 0 0 0 0]
 [0 0 1 0 0 0 0 0]
 [0 0 0 0 1 0 0 0]
 [0 0 0 0 0 1 0 0]
 [0 1 0 0 0 0 0 0]]
```

Then, I would expect students to implement it using procedural approach by adding a separate function for each of the four functionalities of the algorithm. It would give you flexibility in terms of the implementation if later on you decide to replace any one of the four steps with a more specialized implementation. The sample python code using the procedural approach may look like below:

```python
import random
import numpy as np

def Initialization():

  return boards

def Fitness(board):

  return fitness_value
```

COMSATS University, Islamabad

Class: RAI
Subject: AIC530 – Programming for AI
Instructor: Dr. Muhammad Imran

Date: 26-03-24

Genetic Algorithms

```
def Selection(boards):

  return selected_boards

def Crossover(parent_a,parent_b):
  return offsprings

def Mutation(offsprings):

  return mutated_offsprings


def eight_queen_problem():
    boards = Initialization()
    while True:
        fitness = [Fitness(board) for board in boards]

eight_queen_problem()
```

The output I am expecting from the code is the final fit board configuration. Alternatively, you may run the simulation for 10 iterations.