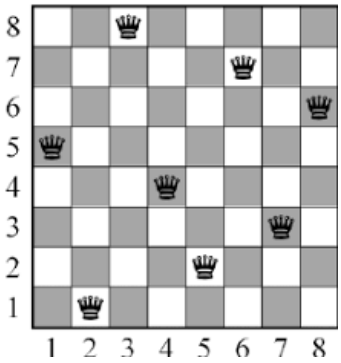
Winter 2019

CS 441/541
Artificial Intelligence

Programming Assignment #2
Due Date: Monday, 2/25 (6.00pm)

The goal of this assignment is to write a genetic algorithm that solves the **8-Queens Problem**.



(*) If needed, please refer to our text/lecture notes for details on the specification of the 8-Queens problem.

Core Elements of the GA:

Initial Population: The initialization contains a randomly distributed population. Lower population size will lead to lots of time in computation of approximate solution. And higher value will cause internal iterations to increase. Therefore choose the population size carefully. Suggestion: try experimenting with population sizes of 10, 100, 500, 1000, etc.

Selection of parents: Just as evolutionary biology requires two parents to undergo meiosis, so does GA. Therefore there is need to select two parents which determine a child; this selection should be done in proportion to the "fitness" of each parent.

CrossOver: Once parents having high fitness are selected, crossover essentially marks the recombining of genetic materials / chromosomes to produce a healthy offspring. Pick a random spot for crossover, and breed two new children (with fitness computed).

Mutation: Mutation may or may not occur. In case mutation occurs, it forces a random value of child to change. Randomly decide whether to mutate based on a **MutationPct**, and if so, mutate one gene.

Your Assignment

Write a GENETIC ALGORITHM (using Python or any programming language you prefer) that solves the 8-Queens Problem.

Your program will start by creating **PopulationSize** random members of the population (including a computation of the fitness function for each one). You will need to devise an encoding scheme for this problem (this should not be difficult). Your program will loop iteratively **NumIterations** times, performing the following functions:

- Randomly select pairs of parents to breed.
- Pick a random spot for crossover, and breed two new children (with fitness computed).
- Randomly decide whether to mutate based on **MutationPct**, and if so, mutate one gene.
- Generate a sufficient number of children to keep the population size constant.

Determining fitness: There are several plausible fitness functions that you can use for this problem – please devise your own and thoroughly describe the fitness function that you are using in your assignment write-up. *Hint*: Determine the maximum number of distinct pairs of queens that can be mutually attacking at any one time and define your fitness function in terms of this quantity.

Selection of parents: One natural way to determine the probability of selecting a particular parent from the general population is to use a "normalized fitness" calculation, e.g.:

$$s_i = \frac{f_i}{\sum_{j=1}^{PopSize} f_j}$$

Where s_i denotes the selection probability of the *i*th element of the population; similarly, f_i denotes the fitness of the *i*th element.

(*) For a stopping condition, you can use a fixed number of generations, or an average fitness criterion.

What to Hand In

After you've written the program, it is up to you to experiment with population size and number of iterations that produces good results. Since not very much happens in a single iteration, you may need a very large number to get a good result, but that is what you will determine.

What to hand in: E-mail Asher (our TA) an electronic version of your code with instructions on how to compile and run it. Please include a short writeup summarizing your findings, technique and conclusions that you reached; be specific about the parameters used for your GA, including the fitness function and mutation; feel free to compare and contrast results using different fitness functions and mutation values.

(*) In your write-up **include a plot** (as seen in lecture) of the average fitness (vertical axis) to population generation number (horizontal axis).

(*) Include **several explicit examples** of individuals sampled from different generations of the execution of your GA (e.g. chessboard examples from initial population, the final population, etc.)

Optional Extensions

There are numerous extensions to this very basic genetic algorithm, as discussed in class. If you are interested in genetic algorithms, I would suggest you consider incorporating your ideas into a final project rather than working more on this particular assignment. But if you'd just like to work on this a little more and still do a separate final project, you can add different mechanisms for crossover, or mutation, roulette or rank fitness, random death (with or without elitism), etc.