Evolutionary Algorithms (89-521)

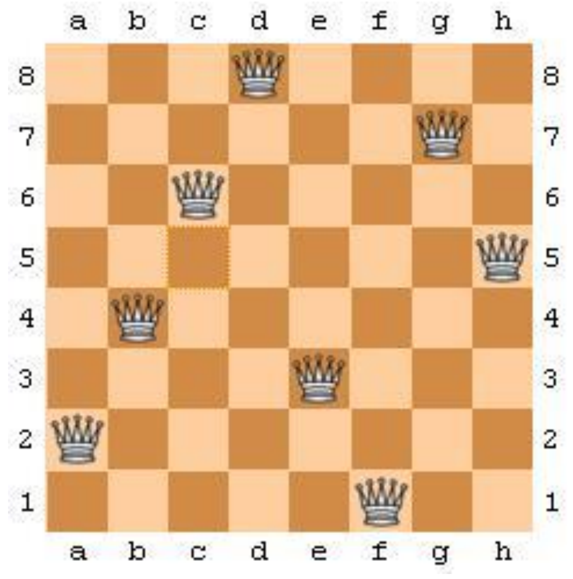
Exercise #1

This project contains 2 problems, which you will solve using Genetic Algorithms (GA).

You must write your own GA code, in any programming language. Although this is not a programming course, and I won't be awfully picky about correct coding (as long as it runs correctly), it doesn't mean you can put all the code in one long main() function! Divide your code into meaningful functions and files.

For each of the following problems, experiment with various combinations of population size, crossover rate, crossover type, mutation rate, and any other parameter you find interesting, and provide the results in your report (which combination worked better, why, etc.). Additionally, for each problem provide a few learning graphs (showing the best and average score for each generation), and of course, provide the best solutions found by GA.

Problem 1: Eight Queen Puzzle



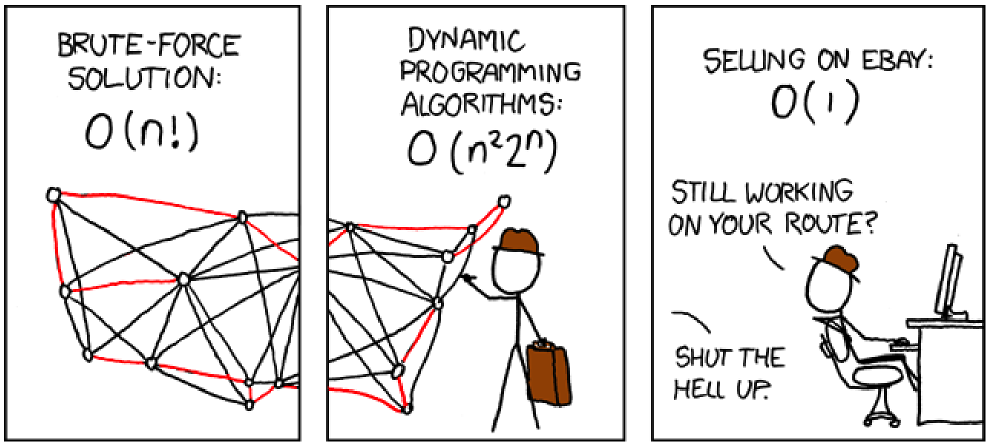
Take a look at the above chessboard. You see that the eight queens are placed on the board, and none of them can capture each other (no two queens are on the same row, column, or diagonal).

The puzzle was originally proposed in 1848 by the chess player Max Bezzel, and over the years, many mathematicians, including Gauss, have worked on this puzzle. The problem can be quite computationally expensive as there are 4,426,165,368 possible arrangements of eight queens on a chessboard (8 x 8 squares), but only 92 correct solutions.

The goal of your GA is to evolve a solution to the puzzle (arrangement of 8 queens such that they don't capture each other). Each chromosome should represent the positions of all 8 queens. Explain how you encode the chromosome. The fitness function should be quite simple.

For comparison, implement a random solution for the problem (brute-force). Compare the performance of GA to that of random search.

Problem 2: Traveling Salesman Problem



In the traveling salesman problem (TSP), a salesman who lives in city A, in a country with N cities, must make a round trip, visiting all other cities in the country, and return home. The goal is to minimize the total distance travelled.

You're all familiar with the TSP problem (you learned approximation methods for it in Algorithms course). And you know that it is an NP-complete problem, i.e., apparently there is no efficient algorithm for solving TSP (which means that any algorithm's worst case running time would be exponential).

Together with this exercise you can find two text files with 'tsp' prefix. Download them. They contain sample data and best solutions for this problem (details below).

The real challenge here is the chromosome representation for the solution (how to encode a route as a chromosome). A straightforward representation would be putting the list of cities one after another in the chromosome (for example, A B C D), but think what would happen when crossing over two chromosomes (for example, how do you cross over chromosome ABCD with ACBD?). You have to make sure each chromosome contains a valid tour (contains all cities, and each city is visited exactly once), even after crossover and mutation. Provide detailed explanation of your chromosome representation in the report.

The fitness function is trivial: just follow the route the chromosome gives and sum up the cost.

At first, generate a few samples to test your program (for example, 5 cities, etc). Imagine each city is on a grid, and assign it a random X and Y value. Calculate the distance between two cities as the Euclidean distance between two points.

When your evolution is complete and you have the best organism, compare the cost of the solution found by GA to the cost of the best solution for the problem. For small problems (e.g., 5 cities), you can just run a brute-force code to find the best solution. By comparing the solution found by GA to the best solution, you can obtain an approximation measure (by how many percentage points the GA suggested route was worse than the real best route).

For the ultimate test, use the data in the file 'tsp.txt'. The file contains the coordinates of 48 cities (each line contains X and Y). You should submit a text file with the best solution you found. The file should contain exactly 48 lines, and one city index per line. For example:

```
32
13
42
1
…
```

Note that the numbers should be from 1 to 48 (not 0 to 47). <u>Since the result will be automatically checked, and deviation from the above instructions, will result in failure of automatic checker!</u>

For comparison, implement a greedy implementation for TSP as follows: Starting from A, at each step move to the closest unvisited city. Compare the performance of greedy approach to best solution. Which approach performs better, GA or greedy?

Again, as in the previous problem, experiment with various GA variables, and include your findings in the report, together with detailed explanation of chromosome representation, several output solutions, and learning graphs.

What to Submit

1. A zip file named 123456789_123456789.zip (replace 123456789 with your ID numbers. If you are submitting alone and not in pairs, then 123456789.zip. If you need to resubmit, add a version number at the end, for example 123456789_123456789_2.zip, etc).
   The zip file should contain:
   - A detailed report (in DOC or PDF) explaining exactly what you did, and the results you obtained. Also include the runtime for different experiments.
   - Two folders containing codes for the two problems above.
2. A text file named 123456789_123456789.txt (or just 123456789.txt if submitting alone) which contains the best result for TSP problem you obtained (see above for exact instructions regarding the format).