**Report about GA when applied over N queen and TS problems**

**by Ephi Frankel and Yarden**

**Part1 – N Queen Problem**

Introduction

A very well known problem in the domain of optimization problems is the N queen problem. Given a board at size N X N, find a placement for N chess queens such that no queen will threat another queen.

In this report we aim to show the results when trying to solve this problem for N=8 with GA. We will also use a randomized CSP that finds an optimal solution as our baseline results comparison.

Baseline approach

There are many possible ways to find a viable solution for the N queen problem. Let us review some of them in terms of speed/performance in order to select the most reasonable as baseline comparison to the GA algorithm.

1. Naïve Brute Force: Generate all possible permutations of N queens at the N X N board for each permutation check if there exists a threat in the board, select a permutation that has no threats.

Complexity analysis: Factorial at N

Algorithm is complete – that is, a solution is promised to be found.

1. Recursive Algorithm: Recursively generate boards that maintain the valid solution throughout their construction (if there is violation, try another path).

Complexity analysis: Exponential at N

Algorithm is complete – that is, a solution is promised to be found.

1. Randomized Algorithm: Generate a random permutation that is a possible solution for the N queen problem such that there is a queen at each row. For each queen from row i: N-2 to 0, move it if it threats any queen at row i+1,i+2, .., N-1, if a valid position found check for the queen at the upper row, if not, re-apply the algorithm, if valid position were found for all rows return the solution.

Complexity analysis: C\*N^2 where C is the number of random permutations needed to find a solution. Note: C is not bounded this applies that the algorithm is not complete, the algorithm will only halt if a solution was found, otherwise, will continue indefinitely. However, if the algorithm halts the solution is optimal. Also, in practice C is usually small.

Since GA remind us of stochastic search in some aspects we have chosen option (3) as our baseline.

Baseline results:

When the baseline approach is applied optimized solution might be found as fast as 0.004 seconds. However, due to the reasons discussed above it is possible that no solution might be found at all.

Fig1 shows an outcome given the baseline method.

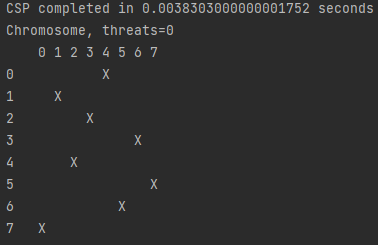


Fig1 – An example for baseline solution.

GA approach

As discussed throughout the lectures GA requires the following main building blocks in order to be applied:

Chromosome: a possible solution of the problem

Fitness function: measures how “fit” is a chromosome.

Selection mechanism: defines how does 2 chromosomes are selected for cross-over.

Cross-over function: defines a re-combination over 2 chromosomes into 2 offspring.

Mutation function: applied over a chromosome causes at a very low probability random change at the chromosome.

Additionally, elitism is a mechanism in which a percent of the strongest chromosomes passed into the next generation at the expanse of the same percent from the weakest chromosomes.

8-Queen GA

First, we consider the following 1-D representation for the problem such that each queen is allocated a single row. To be precise we say that each queen belongs specifically to row and that queens by definition cannot share rows. By this definition we reduce a dimension of the problem since now no 2 queens can ever be located on the same row, so for each we define the location of at row . So, we have that as our representation.

Next, we consider that then each can be represented with exactly 3 bits (000 = 0, 001 = 1, …, 111 = 7). Since this is the case we define the Chromosome to be such that is the bit representation for the location of , is the bit representation for the location of , …, is the bit representation for the location of .

**Part2 – TS Problem**