Course on Artificial Intelligence and Intelligent Systems

Exercise for Evolutionary Computing

Henning Christiansen
Roskilde University, Computer Science Dept.
 Version 2 nov 2005

1 Remarks

This exercise is written down for documentation and reference following the course day 1 nov 2005 where it was introduced on the blackboard.

2 The exercise

We consider the problem of placing n queens on an $n \times n$ chess board in such a way that no queen threatens another; for practical reasons, we let $n = 5$. The following is an example of a solution.

Q				
			Q	
	Q			
				Q
		Q		

We refer to columns and rows of the board according to its orientation on paper. For the genetic programming exercise, we define as chromosome any checkerboard with exactly one queen in each column, and the fitness function f is defined as the number of queens in a given chromosome (board configuration) that are not threatened plus one.[1] The following chromosome has a fitness of 2 as only one of the queens is not threatened.

Q				
			Q	
	Q			Q
		Q		

[1] In the first formulation of the exercise, we did not add one in the fitness function. As a result, the evolution process tended to degenerate as it is often the case that all queens are threatened, and a chromosome of fitness zero cannot be selected. Thus the process often leads to having only one parent chromosome to be selected as parent, or even to populations with all zero fitness chromosomes.

Crossover for two boards is defined be selecting at random a horizontal split of the board, i.e., into 1+3, 2+2, or 3+1 columns, and then swopping the parts to form the new individuals. Assume the following parameters:

- Population size is set to 6.
- Crossover probability of 0.7.
- Mutation is controlled by selection one random column in one random board, and with a probability of 0.5 either doing noting or changing the position of the queen in that column in a random way.

Now the exercise goes a follows. Draw 6 empty boards for the first generation, and put in queens in each column in a random way. Now use the standard evolution algorithm to derive new generations and stop when either you have found a perfect solution (with fitness 7), or you have got tired of the exercise. In the last case, make you own evaluation of whether the process seems to show some sort of improvement, or, if this is not the case, how the parameters might be adjusted.

To make random choices, you may use dices, coins, a table of randomly generated numbers, or your own inventions.

3 Programming project

Implement it.