Course on Artificial Intelligence and Intelligent Systems

Exercise for Evolutionary Computing

Henning Christiansen
Roskilde University, Computer Science Dept.
 Version 10 nov 2008

Introduction

The purpose of this exercise is to get an understanding of how evolutionary algorithms work by hand simulation. The example is chosen in a way which makes the translation from chromosome to the object it encodes trivial, and the fitness function is simple to calculate by hand. However, the example is not representative for the class of problems where evolutionary algorithms are most appropriate. This approach is best suited for optimization problems where there are no other good methods, and the n-queens' problem that we use below is not of that kind.

Often cited applications concern design of physical system such as antennas, or electronic circuits, especially analogue ones, which should be optimized to approximate a given mathematical function as good as possible and other properties such as response time, etc.

1 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		

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.

[1] In an earlier formulation of the exercise, we did not add 1 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.

Q				
			Q	
	Q			Q
		Q		

Crossover for two boards is defined by selecting at random a vertical split of the board, i.e., into 1+4, 2+3, 3+2, or 4+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 nothing or changing the position of the queen in that column in a random way.

Now the exercise goes a follows. Draw 6 empty boards on a piece of paper 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 6), 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.

2 Important question

The advantage of this example is that it is very easy to do by hand, but as an example of evolutionary computing, it is not very good. In fact, evolutionary computing is not very well suited for this particular problem. Explain why!

Can you indicate properties of a good genetic encoding and cross-over operator?

3 Programming project

Implement it. Or better, find an application for which evolutionary computing is better suited, and implement that. There are available software packages that may make the job easier.