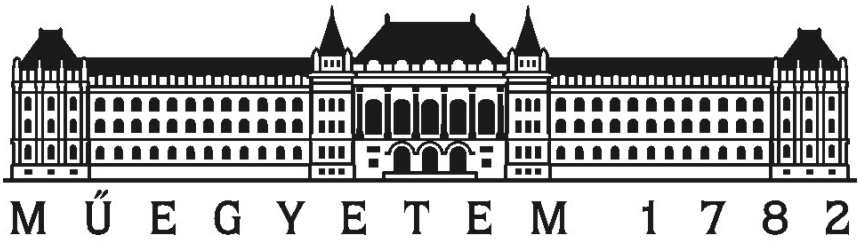


Evolutionary Algorithms

Homework

Eighth task

Author
Carlos Sánchez Páez
http://www.github.com/csp98

Budapest University of Technology and Economics

Academic year 2018-2019

1. Design a genetic algorithm to solve the n-queens problem (you don't have to find every layout, just one where not pairs hit eachother). What sort of crossover and mutation operators would you use? What is the best criterion for stopping? What is a suitable fitness function?

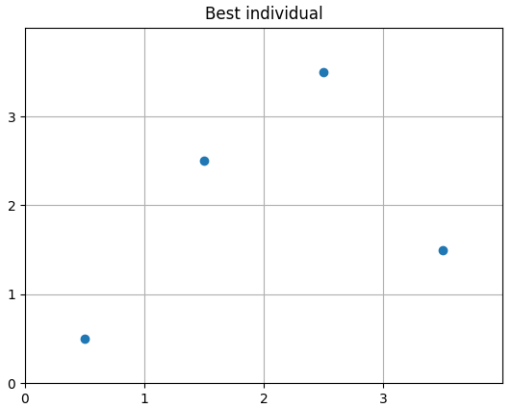
   For this problem we will use a permutation representation in the following way:
   Each permutation index will represent a column and each element, a row. Let's see an example:

   $$[0,2,3,1]$$

   This permutations means that the positions of the queens will be:

   - (0,0)
   - (1,2)
   - (2,3)
   - (3,1)

   Which gives us the following board:



   As this is an adjacency problem, we need to use the apropiate operators for crossover and mutation. I chose PMX (partially mapped crossover) crossover and inversion mutation.

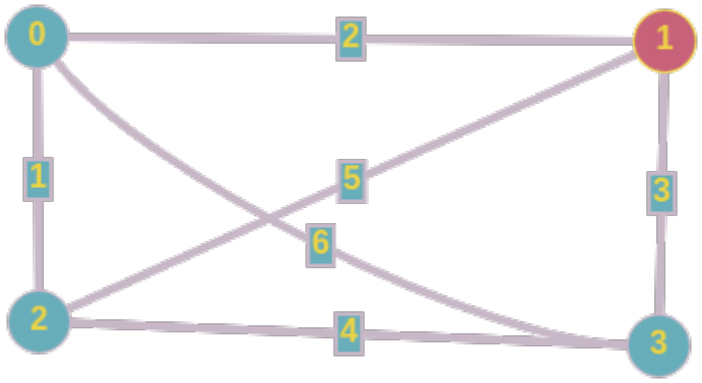
   The fitness functions defines as follows:
   The queens won't be on the same row or collumn (they are permutations without repetition, so $p_1(x) \neq p_2(x) \wedge p_1(y) \neq p_2(y)\ \forall p_i \in P$ with P as the set of all possible permutations).
   Thanks to this representation, we don't have to worry about vertical or horizontal attacks between queens: the only possible attacks are the diagonal ones. This means that our fitness function will be the number of queens in a diagonal, if there is more than one.
   This makes our problem a minimization one: the permutation with fitness= 0 is the best one, as there are not attacking queens.
   As this problem is a CSP one, we should stop when we find the correct layout (fitness= 0), which is the feasible one.

2. Find a new example (that is we haven't mentioned in the class before) for a problem, that can be solve with genetic algorithm using permutation representation. What is the type of it? (CSP-FOP-COP, scheduling-adjacency)? What would be a suitable fitness function?

   The **Minimum Spanning Tree** problem consists on finding a subgraph on a given graph will connects all the vertices, without cycles, with the minimum possible total weight. It is a COP problem, as we have constraints (all the vertices must be reached) and an objective function to minimize (the sum of the weights of the edges).
   The permutation representation of the problem should be the following:

   - Each elements represents an edge.
   - 0 means that the edge is not taken.
   - 1 means that the edge is taken.

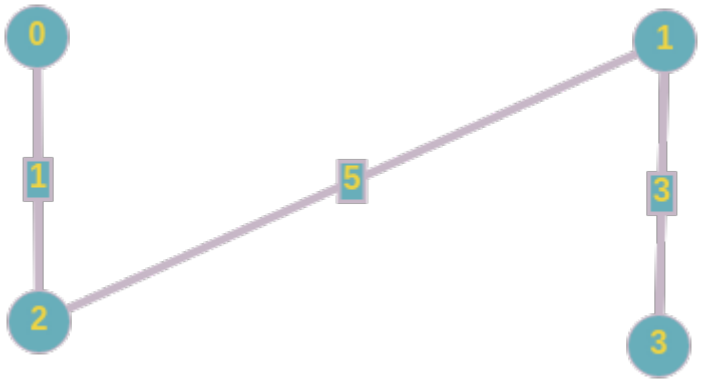
   For example, if we have the following graph:



   The permutation

   [1,0,1,0,1,0]

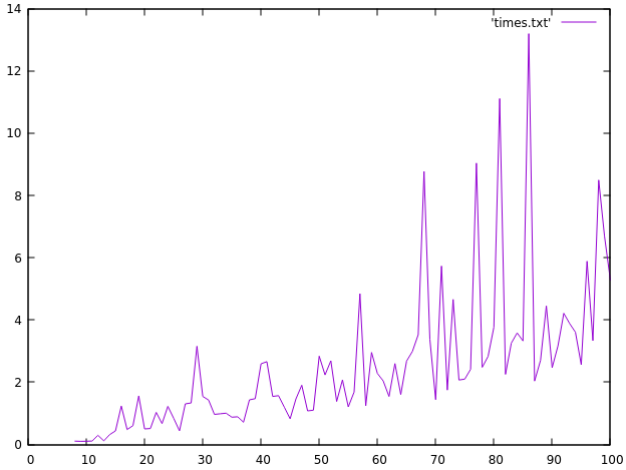
   means that the chosen vertices will be 1,3 and 5, so the spanning tree will be:



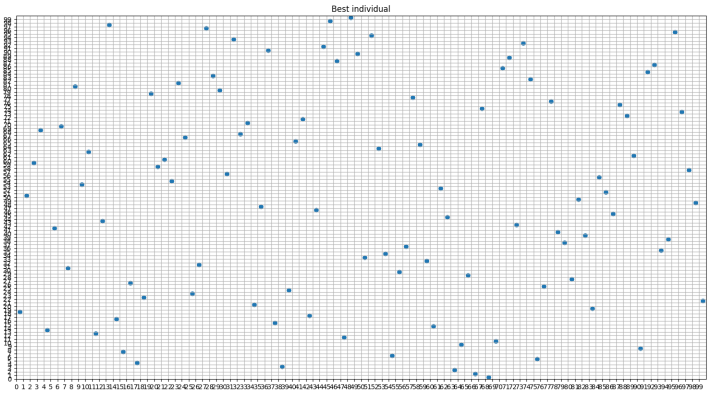
   Our fitness function should be the sum of the weights of the edges, so we have to minimize it. We also have to check that the individuals are feasible (all the vertices must be reached).

3. Write a program you designed in the first exercise, and examine the running time. What is the maximum size of the chessboard your function can find a good layout? Try different fitness functions (that is different penalties).

   My program could find a good layout for 100 queens in 5.28 seconds. It needed 141 generations. However, it could reach bigger sizes.
   The running times obtained are represented in the following plot:



   Here is the layout with 100 queens:



Bibliography

[1] Course Webpage
`http://math.bme.hu/ safaro/evolalgen.html`

[2] `https://tex.stackexchange.com/`