UNIVERSIDAD D CÓRDOBA

Session 2

Metaheuristics

Academic year 2020/2021

This second lab session is oriented to the development of genetic algorithms with different representations. The session is divided into two parts: Binary encoding; Integer encoding. The student must carry out everything that is requested in this document, and to generate a detailed report in which the answers and analyzes carried out are justified. Both **the new code and the detailed report must be submitted to Moodle before the deadline**, taking into account that this second session lasts 3 sessions.

1. *Binary encoding*. In this first section we are going to work with a binary representation of the knapsack problem to solve it through a generational genetic algorithm. The (incomplete) Python code is in the Kanpsack.py file. The binary representation [1, 0, 1, 1, 1] indicates that objects 0, 2, 3, and 4 have been chosen. Analyze the code and verify that it is correct and meets the requirements. Then complete the code (only *GeneticOperators* function). Use the crossover and mutation operators of your choice. You can vary the number of objects, as well as the weights and prices to check that your algorithm works.
   Answer the following questions in a justified manner and with the help of graphs when necessary:
   - How does the algorithm behave as we modify the number of solutions, generations, tournament size, crossover probability, and mutation probability? Add more objects to the problem to check the behavior of the algorithm.
   - Do you always get the best solution? Why? What does it depend on?
   - Modify the code to incorporate elitism. The best solution is kept in the elite and it is never lost until a new better one is obtained. This solution is the one that is returned at the end. Have you managed to improve? Why?
   - So far, we have started with valid solutions. Change it so that any solution can be generated. Does it affect the final performance? Analyze this issue.
2. *Integer encoding*. In this second section we are going to work with an integer representation of the knapsack problem through a generational genetic algorithm. Now each object can appear more than once so the binary representation [1, 0, 1, 1, 1] is no longer valid. Use a representation [0, 2, 0, 1, 3] to indicate that you have 2 objects of type 1, 1 object of type 3 and 3 objects of type 4. Use the code from the previous section to adjust it to the new problem and representation. You can vary the number of objects, as well as the weights and prices to check that your algorithm works.
   Answer the following questions in a justified manner and with the help of graphs when necessary:
   - How does the algorithm behave as we modify the number of solutions, generations, tournament size, crossover probability, and mutation probability? Add more objects to the problem to check the behavior of the algorithm.
   - Do you always get the best solution? Why? What does it depend on?

- Modify the code to incorporate elitism. The best solution is kept in the elite and it is never lost until a new better one is obtained. This solution is the one that is returned at the end. Have you managed to improve? Why?
- So far, we have started with valid solutions. Change it so that any solution can be generated. Does it affect the final performance? Analyze this issue.