Multi-Criteria Optimization and Decision Analysis
for Embedded Systems Design (MCODA)

Lab 5-6: Metaheuristics - From single to multi-objective optimization

Exercise 1: Travelling salesman problem - Simulated annealing

Download the dataset for the cities of Djibouti at `http://www.math.uwaterloo.ca/tsp/world/countries.html`.

Implement a simulated annealing algorithm to solve the TSP.

(a) Model a tour as a graph (hint: have a look at the `networkx` library `https://networkx.github.io/documentation/stable/`)

(b) Start with a randomly-generated tour

(c) Consider that a new solution is a random swap of two edges in the graph

(d) Apply the simulated annealing methodology (probability of accepting the new solution, updating the temperature, etc.) to perform the optimization

(e) Consider a starting temperature of 10000, an ending temperature of 1, and a linear cooling rate $\in [0.001, 1]$

(f) Play with the parameters and observe how the algorithm behaves

Exercise 2: Travelling salesman problem - Genetic algorithm

With the same Djibouti dataset, implement a genetic algorithm to solve the TSP.

(a) Model a tour as a vector

(b) Consider a roulette wheel for the selection operation; try also a tournament selection

(c) Consider a single-point crossover operation (hint: remember that a city can only appear once in the tour!), $p_c \in [0.7, 0.95]$

(d) Consider a random swap for the mutation operation, $p_m = 0.3$ (probability for an offspring to mutate); alternatively, you can consider the mutation of each component of the vector ($p_m \in [0.001, 0.01]$)

(e) Consider a constant population size of 100; for the replacement strategy, rank the solutions and sample a 100 individuals from the merged population

(f) Apply the genetic algorithm methodology to perform the optimization

(g) Play with the parameters and observe how the algorithm behaves

(h) What if you only select 90 solutions in the replacement strategy and complete the population with 10 randomly-generated solutions?

Exercise 3: Flowshop scheduling problem - MOO (towards NSGA-II)

m pieces have to be processed through n machines, each of them requiring a given duration on each machine. The problem is to find the optimal schedule (order of operation) in order to minimize the makespan (time when the last piece is processed, that is, the total completion time) and the total weighted tardiness (sum of tardiness for each piece, considering their relative weight: $\sum_{i=1}^{n} w_i \max(T_i - D_i, 0)$, where T_i is the time when the piece i is processed and D_i is its due time).

Download the file (`fsp.txt`) from Moodle. It contains information of an FSP instance where 200 pieces have to be processed by 10 machines (denoted M_j). Each of the 200 rows in the file represents the information for a piece. For each piece, there are 12 columns. The 2 first columns represent respectively the weight and the due time of a piece, for the computation of

the total weighted tardiness. The 10 remaining columns represent the processing time by each machine $M_1, M_2, \ldots, M_{10}$. To simplify the problem, we will consider here that the order for the machines is fixed and given by the order in the file, so only the order for the pieces has to be optimized.

(a) Model a schedule as a permutation

(b) Write a function to compute the makespan and the total weighted tardiness

(c) Consider a single-point crossover operation (hint: remember that a piece can only appear once in the schedule!), $p_c \in [0.7, 0, 95]$

(d) Consider a random swap for the mutation operation, $p_m = 0.3$ (probability for an offspring to mutate); alternatively, you can consider the mutation of each component of the vector ($p_m \in [0.001, 0.01]$)

(e) Use the provided library to filter the Pareto optimal solutions from a population and find their Pareto ranks

(f) Consider a constant population size of 100; for the replacement strategy, consider the Pareto rank of the solutions (note: if, for instance, the rank 2 would make your population size bigger than 100, select them randomly in order to achieve 100)

(g) Apply the multi-objective genetic algorithm methodology to perform the optimization

(h) Play with the parameters and observe how the algorithm behaves

(i) What if you only select 90 solutions in the replacement strategy and complete the population with 10 randomly-generated solutions?

Remark: This procedure is not exactly NSGA-II as it does not consider the crowding distance concept. It however contains the Pareto rank concept which should give you enough understanding of how a multi-objective version of a genetic algorithm would behave.

Exercise 4: Travelling salesman problem - DEAP

DEAP is a Python algorithm allowing to easily implement the different block of evolutionary algorithms. Follow the tutorial at `https://deap.readthedocs.io/en/0.9.2/tutorials/index.html` to get familiar with it.

Use DEAP to implement a genetic algorithm for the TSP.

Exercise 5: Flowshop scheduling problem - DEAP

Use DEAP to implement an NSGA-II algorithm for the FSP.