Neural and Evolutionary Computation (NEC)

A2: Optimization with Genetic Algorithms

Objective

Implementation of a genetic algorithm (GA) to find the solution to the Job Shop Problem.

The Job Shop Problem or the Job-Shop Scheduling Problem (JSSP)

The Job Shop Scheduling Problem (JSSP) is a well-known, classical combinatorial optimization challenge in the field of operations research. In this problem, there are multiple jobs that need to be processed on multiple machines. Each job consists of a series of operations, and each operation must be processed on a specific machine for a given amount of time.

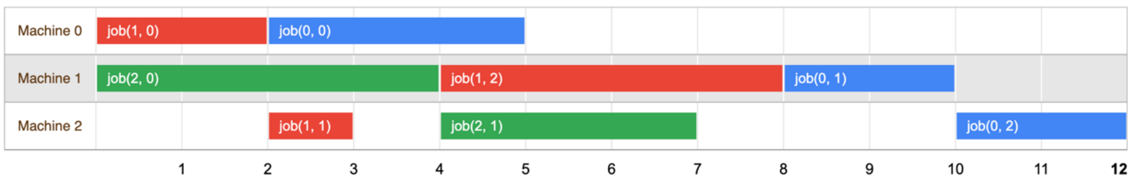
There are several constraints for the job shop problem:

- No task for a job can be started until the previous task for that job is completed.
- A machine can only work on one task at a time.
- A task, once started, must run to completion.

Example problem: Below is a simple example of a job shop problem, in which each task is labeled by a pair of numbers (m, p) where m is the number of the machine the task must be processed on and p is the *processing time* of the task — the amount of time it requires. (The numbering of jobs and machines starts at 0.)

- **JOB 0** = [(0, 3), (1, 2), (2, 2)]
- **JOB 1** = [(0, 2), (2, 1), (1, 4)]
- **JOB 2** = [(1, 4), (2, 3)]

Possible solution of this problem:



You can get a more detailed explanation from:
https://en.wikipedia.org/wiki/Job-shop_scheduling
https://developers.google.com/optimization/scheduling/job_shop

Genetic algorithm

Given a matrix of tasks like the one we have seen previously, the objective is the implementation of a genetic algorithm to obtain the distribution of the tasks in the machines that minimizes the total computational time of the task.

For the activity, you will have to take into account these points:

- How to codify the problem into a chromosome for the genetic algorithm
- How to validate that a chromosome represents a valid solution
- How to compute the total execution time of a solution (fitness of the solution).

In this activity you will also need to think about how to select the appropriate methods for crossover and mutation that will work with your description and codification of the problem.

Data and libraries

There are many examples of datasets that can be used. For instance, you can use this link which contains some of the most common datasets studied for the problem:

http://people.brunel.ac.uk/~mastjjb/jeb/orlib/files/jobshop1.txt

You can use other datasets available form the internet that have the same structure for the problem.

Optional part: Implementation of a second optimization algorithm (2 points)

If you want to obtain the full grade, you should implement a second optimization method to try to reach the best possible solution. In this case you can choose any of the methods that we have seen in class (tabu search, simulated annealing, extremal optimization, …).

In this case you should be able to reuse most of the code done for the Genetic Algorithm (codification of the problem, total time computation, etc…), so you can focus on the implementation of the other method.

Delivery of the activity

This assignment can be done alone or in pairs (groups of two)

For this activity you must deliver **one PDF document** that includes:

- A link to the Github repository where the code of all the activity is accessible. More details on the code in the following sections. Remember that we require a history of all the commits, not only the last one with the final code.
- The name of the file should be **A2-Name1Surname1-Name2Surname2.pdf**

Also, in the document you must include the following two sections:

1. **Description of the chromosome, and adaptations done to the algorithm:** Explain with detail how you translate the problem into the chromosome, and the different techniques that you have implemented for selection, mutation and crossover. You must AT LEAST implement two different techniques for each of those. You can also use elitism in the code if you think.

   You must also explain how you choose the size of the population and how you identify that the system reaches a stationary state.

2. **The results of executing the code for 3 problems of different sizes**: We recommend running at least one problem with 3-5 machines, one with around 10 and one with 15 or more machines. Optionally you can try with even larger problems, since we will consider the complexity of the selected datasets.

   For each of the executions you should include in the document:

   a. The description of the dataset (URL where the data was collected, explanation of the data).
   b. The results obtained with at least 6 different combinations of parameters described in the previous section.
   c. For the best solution, present a figure with the evolution of the minimum total traveling distance in each of the algorithm steps.

3. **In case you implement another optimization method**, you should describe the method used, the parameters of the method (what parameters have been used and how they have been chosen), a comparison of the results obtained versus the genetic algorithm.