Practical Machine Learning (ITF31519)
Portfolio/Final Project

Submission deadline: 14th December 2021 at 11:59 AM.

Instructions:

1. There are **TWO** exercises. You must implement and write a report for BOTH of these exercises.
2. The portfolio is **individual**. No cooperation of any kind is allowed.
3. Programing Language: Python. You are allowed to use any library of your own choice.
4. Your submission will be atomically checked for plagiarism. In case of any plagiarism, it will be handled by Østfold University College´s policy. You can find more about it at https://www.hiof.no/english/studies/examinations/cheating-and-plagiarism/

Materials to Submit:

1. **Coding**: Complete coding in a state that it can be run to produce the result. The two exercises should be in TWO different folders.
2. **Report:** A SINGLE report for both exercises. The report (at most FIVE A4 size page in 10 size font) must contain your responses to the points mentioned in each exercise. The responses must be based on your implementation and achieved results.

Submission: Put everything in a zip folder named using the following convention: FirstName_LastName.zip and upload in canvas.

Submission place: Canvas

Part-1

Implement a two-layer perceptron with the backpropagation algorithm to solve the parity problem. The desired output for the parity problem is 1 if an input pattern contains an odd number of 1's; otherwise, the output is 0.

Use a network with four binary input elements, four hidden units for the first layer, and one output unit for the second layer. The learning procedure stops when an absolute (difference) of 0.05 is reached for every input pattern.

Other implementation details are:

- Initialize all weights and biases to random numbers between -1 and 1.
- Use a logistic sigmoid with a = 1 as the activation function for all units.

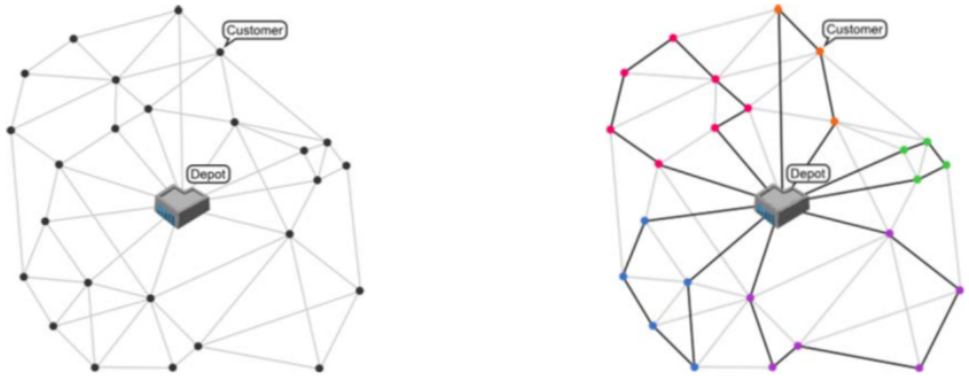
After programming is done, vary the value of learning rate from 0.05 to 0.5 with increment 0.05, and report the number of epochs for each choice of learning rate.

You should use proper graph to analyze the implementation and the relationship between the learning rate and the the number of epochs.

Part-2

In this part, you will implement genetic algorithm (GA) to solve a NP-hard combinatorial optimization problem ─ The Multiple Depots Vehicle Routing Problem (MDVRP).

Vehicle routing problem (VRP) is a classical combinatorial optimization. VRP formulations are used to model a vast range of issues in many application fields, transportation, supply chain management, production planning, and telecommunication, to name but a few. A typical VRP can be stated as follows: a set of geographically dispersed customers with known demands are serviced by a homogenous fleet of vehicles with limited capacity. Each customer is to be fully serviced exactly once, and each vehicle is assumed to start and end at the same depot. The **primary objective is to minimize the total distance travelled by all vehicles**. However, in a large number of practical situations and to satisfy real-life scenarios, additional constraints are usually defined for variants of the VRP.



(a) (b)

Fig. 1: A hypothetical instance of a VRP (a) and its solution (b)

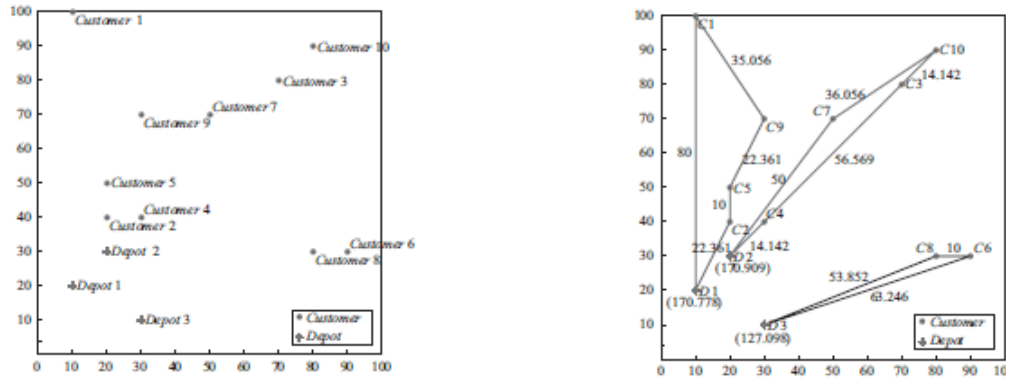
In this project, you need to solve the multi-depot VRP (MDVRP), which is an extension of the classical VRP except that there are multiple depots with many vehicles, and every customer should be serviced a vehicle based at one of several depots. Like the typical VRP, in the MDVRP, every vehicle route must start and end at the same depot. **The challenge is to make a schedule for each vehicle individually** so that the vehicles drive in the most efficient way, optimizing one or several objectives.The MDVRP is NP-hard, which means that an efficient algorithm for solving the problem to optimality is unavailable.

Problem Formulation:

Formally, the MDVRP can be defined as follows. We are given a set of depot locations and a set of customer locations, which are assumed to be disjoint (even if two points share the exact physical coordinates, they are still handled as different entities). Each customer is characterized by their own demand. A fleet of vehicles with limited capacity is based at each depot. Each vehicle originates from one depot, services the customers assigned to that depot and returns to the same depot. The MDVRP consists of determining the routes for multiple depots with multiple vehicles per depot in parallel. Further, each depot has a set of customers. The route should also optimize predefined objective(s) as well as satisfy the following conditions:

(i) every customer appears on exactly one route.
(ii) every route starts and ends at the originating depot.
(iii) capacity limit: the total demand of the customers on any route does not exceed a vehicle's capacity.

(iv) route limit: the total duration of a route does not exceed a preset value (**for this project, it is only for those problems for which this value is mentioned in the test data**).



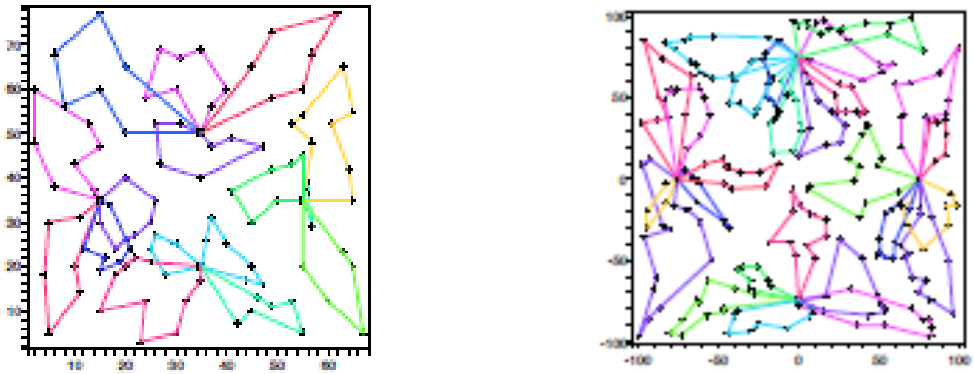
(a) (b)

Fig. 2: A hypothetical instance of a MDVRP: (a) locations of customers and depots, (b) one solution

Fig. 2 presents an example for MDVRP with an arbitrary solution. Generally, the objectives of the MDVRP are to minimize the total combined route duration for all required vehicles from all depots, minimize the time spent in serving all customers, or minimize the number of vehicles needed in serving all customers. In this project, **your main goal is to minimize the total distance travelled by all vehicles across the depots to serve all customers**.

Algorithm

As mentioned earlier, to solve the MDVRP, you need to implement the genetic algorithm (GA) that has been already discussed in lectures. In order to get the optimal/near-optimal results, you may check several forms of representation, genetic operators, and selection mechanism. It would be beneficial to test whether *elitism* gives a better solution.



(a) Solution for a test problem with 100 customers, depot no. 2,3,4 have the capacity limit of 100

(b) Solution for a test problem with 249 customers, 4 depots with capacity limit of 500 and route limit of 310

Fig. 3: Solution representation

Note that, GA parameter values (population size, generation number, crossover rate, and mutation rate) are correlated and your GA will successfully find the optimal values if you use appropriate parameter values. However, there is no definite rule to find the appropriate parameter values. Therefore, you should test different set of parameter values to decide the appropriate values.

Things To Do

To test your code, we uploaded several benchmark test data and their solutions. The description of problem and solution data file formats is also included. Y**our code must have the option to read the test data according to the given format**.

Since the MDVRP is an NP-hard problem, and GA is a heuristic algorithm, your implementation may not achieve the optimal total distances mentioned in the solutions for the test problems. It is not uncommon, and we are aware of this. While testing your implementation, we will accept if your produced solutions are within 15% of the known optimal values mentioned in the uploaded solutions.

Note that, your implementation must produce the solution strictly following the text format as output (Fig. 4, also mentioned in the provided solution data files).

```
576.87
1  1  60.06  71  0 44 45 33 15 37 17 0
1  2  66.55  79  0 42 19 40 41 13 0
1  3  47.00  78  0 25 18 4 0
2  1  53.44  73  0 6 27 1 32 11 46 0
2  2  79.47  80  0 48 8 26 31 28 22 0
2  3  81.40  77  0 23 7 43 24 14 0
2  4  23.50  54  0 12 47 0
3  1  50.41  75  0 9 34 30 39 10 0
3  2  25.22  54  0 49 5 38 0
4  1  47.67  67  0 35 36 3 20 0
4  2  42.14  69  0 21 50 16 2 29 0
```

Fig. 4: Expected text format for your solution

Report:

You should write a report answering the points below.

1. Describe the Chromosome representation that you used in your implementation. Also, mention another representation that could be used for this problem and why this representation is also suitable. Defend your choice of the most suitable representation.
2. Describe whether the crossover and mutation operators will produce infeasible off-spring(s) after executing. If yes, how did you handle that? If not, why not?
3. In GA, the parameter values (population size, generation number, crossover rate, and mutation rate) often have some form of relationship. Considering that you tested several sets of parameter values, what relationship did you observe among these parameter values?