# 2018 Project proposal

## An Electric Vehicle Routing Problem with limited charging capacity at stations

The considered decision problem consists in planning a freight transportation service provided by Electric Vehicles (EVs). A fleet of identical EVs (*NumV* is the number of vehicles) is available to serve a set of customers. The vehicles' speed (*Speed* [km/h]), capacity (*Capacity* [kg]) and battery capacity (*Batt* [kWh]) are given. The energy consumption rate for km (*Ro* [kWh/km]) is constant. Note that all distances are given in kilometers [km] and all times in hours [h].

There is a single depot from which all vehicles must depart and return at the end of their route. Each vehicle can perform a single route. Each customer must be served by a single vehicle and it is characterized by an identifier (Id), a demand (*Q* [kg]), a service time (*ST* [h]) and a time window [*R*, *D*], where *R* is the release date and *D* the deadline for starting the service (both in [h]).

If needed the vehicles can stop in a set of recharging stations during their routes. At recharging stations the battery is recharged at a rate (*Alpha* [kWh/h]), so the final charge depends on the time spent by the vehicle for recharging. Partial battery recharges are allowed. Each station has a limited number of chargers (*NC*): therefore at most NC vehicle can recharge at the same time. If a vehicle must recharge at a station but there is no available charger then the vehicle can wait at the station. Recharging requests are served in FIFO order.

The vehicles must serve all the customers with routes whose maximum time duration is given (*Tmax* [h]). The objective is to minimize the total distance traveled.

Data instances are provided as text files, each including the same set of data tables. Each table is specified by a line in the text file with the table name followed by the header of the columns and then by a set of rows with the table data. In particular, the following tables are included in an instance file:

- **Nodes table**, specifiying for each node the identifier, the x and y coordinates in a Euclidean space, and the node type (D=depot, C=customer, S=station)
  An example is
```
NODES
   Id          x             y              Type
    0      25.00         25.00                 D
    1       6.35         13.85                 C
   31      12.50         12.50                 S
```

- **Rechargers table**, specifying the number of recharging plugs available in each station. The available recharging plugs will range from 1 to 3, so a plug in a station can be univocally identified by a progressive number ranging from 1 to 3.
  An example is
```
RECHARGERS
   Id      NumRec
   31           1
```

- **Customers table**, specifying for each customer the identifier, the demand (Q), the service time (ST) and the time window ([R, D])
  An example is
  ```
  CUSTOMERS
     Id          Q            ST            R                D
      1         28          0.28         5.40             9.19
      2         27          0.30         3.89             8.27
  ```

- **Global instance data table**, specifying a set of data for the instance, i.e., the maximum time for tour (Tmax), the number of vehicles (NumV), the capacity of vehicles (Capacity) the speed of vehicles (Speed), the unitary cost for distance (DistCost), the battery capacity of vehicles (Batt), the battery consumption rate for distance unit (Ro), the battery recharge speed for time unit (Alpha).
  An example is
  ```
  INSTDATA
   Tmax        NumV      Capacity Speed        DistCost  Batt    Ro Alpha
     20           3           500    50               5    24  0.15  6.00
  ```

**The project**

The purpose of the project is to design a metaheuristic algorithm that is able to find good solutions in a reduced computation time. The algorithm must be coded in one of the following languages C++, C# or Java. The executable must run on the teacher's notebook (Windows 10, 64 bit). The program (called for example EVRPC.exe) must accept as input on the command line the following data:

EVRPC.exe <*data input file*> <*Max allowed time in seconds*>

The input file, if not differently specified with a path, must be located in the same folder of the executable.

A set of instances will be soon made available on Aulaweb.

Three different maximum time limits must be tested: 10 seconds, 30 seconds and 60 seconds. The maximum time is the limit for the algorithm computation NOT including the time for loading the data from the input file.

The executable must:

1. produce as output a text *InstanceName_Result.csv* file containing the produced solution with the following format (each information in a line must be separated with a ";"):
   - a line reporting the objective (e.g., `Obj;1773.824`);
   - a line reporting the CPU time (e.g., `Time;8.34`);
   - an header line as the following:
     `Vehicle;Orig;Dest;Dist;Rel;Dead;ServTime;DemDest;ArrTimeDest;ArrBattDest;ArrLoadDest;RecTime;PlugUsed`

- a set of lines, one for each arc in the solution, reporting (according to the header) the following data separated by ";": the vehicle identifier, the node origin of the arc, the node destination of the arc, the arc distance, the release date of the destination node, the deadline of the destination node (note that for depot and stations these data can be fixed equal to zero), the service time of the destination node (if it is a customer node, zero otherwise), the arrival time at the destination node, the level of battary on the arrival at destination node, the vehicle load on arrival at the destination node, the time spent for recharging (if the destination node is a station, zero otherwise), the id of the used charger plug (1, 2 or 3).

2. Produce (in append) a file *output.csv* in which each row contains (separated by semicolons) the data for each run executed, i.e., the input file name, the input time limit (seconds), the CPU time actually used to find the solution (seconds) and the objective. An example is: `Instance_1;10;10.23;1098.25`

Note that, assuming that the implemented algorithm makes use of random variables, for each instance 5 runs must be executed and the evaluation will consider the average results over 5 runs.

The final project delivery must include:

1. the source files (specifically, the project environment, e.g., Visual Studio or Eclipse);
2. an executable that must run on the teacher's Windows 10 notebook;
3. for each instance the best solution file found (*InstanceName_Result.csv*);
4. output.csv
5. a document or slide presentation describing in details the implemented algorithm. You will use the slides to describe your work after your final delivery.

**Evaluation**

The project will be positively evaluated (27/30 mark) if it runs producing feasible solutions and if you produce all the deliverables required. Soon a solution verifier application will be available and shared on Aulaweb.

The algorithm quality will be evaluated on the basis of the produced results:

- 30/30 cum laude to the student team producing the best average results for all the three different time limits.
- 30/30 to the student team producing the best average results for at least one time limit but with an average percentage deviation not exceeding 2% with respect to the best results for the other time limits.
- 29/30 to the student team whose average results will have a percentage deviation not greater than 5% to the best average results for all the time limits.

- 28/30 to the student team whose average results will have a percentage deviation not greater than 20% the best average results for all the time limits.

The student teams must complete the project by one of the two following due dates:

- 1/3/2019
- 15/9/2019

The project will be evaluated taking into account the results of all the projects delivered up to the due date, that is, on the basis of the current best results. Example: if teams A and B deliver their work by 1/3/2019 they will compete for a 30/30 (possibly cum laude), and the best group (say A) fix the current best results. If groups C and D deliver by 15/9/2019 they not only compete each other but also with the current best result of group A.

**Final notes**

This project proposal is presented in the class of November 21st 2018. The students that accept to work on this project must communicate their decision to me by December 19th 2018 (last class). The students can work also in teams composed by at most 2 persons. **This is the only call for projects.**

If any team believes that its own computer could be too obsolete for providing high performance results, a fair test setup can be used, i.e., all the experimental tests can be performed on a Windows server computer. This means that (1) all the teams must provide me with the executable <u>before the chosen due date</u> (2) I will run with a batch all the tests and (3) I will provide back the results to the groups for preparing the required delivery.