Case Study
Load balancing of a bike fleet

(**Professor in charge** : Éric Soutil (eric.soutil@lecnam.net

1) **Objective**

The objective of this project is to study an optimization problem. You will have to propose a model and then solve instances of the problem. This problem concerns the management of a self-service bicycle (SSB) fleet. Several formulations of this problem are possible. We suggest a quadratic formulation using integer and 0-1 variables, but every other valid formulation will be accepted, in particular you may propose a more classical linear formulation, inspired by the capacitated traveling salesman problem (TSP). If you choose a quadratic formulation, you will have to apply a linearization technique, presented during the course. You will also have to use the Julia programming language, its library JuMP and a free ILP solver (Cbc) for solving small instances and to implement a heuristic method to solve large instances..

2) **Description of the project**

A company is in charge of a self-service bicycles (SSB) fleet in a big city. A great deal of stations are installed in the city. A user may take a bike that is available at a station A and go to a station B where he drops the bike taken at A. Some locations of the city are more attractive than others Therefore the number of bikes that are available at each station fluctuates a lot during the day. This is a double problem : a station may be empty (no bike available) if it is not attractive or located high up in the city ; on the contrary, in the a touristic area, a station may be full (each station can contain a limited number of bikes). This is naturally problematic since a biker arriving at a full station must find another station. To avoid these two problems as much as possible, the company in charge of the fleet balance the stations at night : a trailer circulates in the city, visits the different stations, unblocks the saturated stations and deposit some bikes at the too empty stations.

The objective of this project is to determine the tour of the trailer that balances the bike stations.

- **Hypotheses and contrainsts that must be respected** : the tour of the trailer begins at the wharehouse where a stock of bikes, sufficiently important, is always present. The trailer is loaded with a given number of bikes (this number will be a variable of the problem, it is not fixed). The the trailer must pass through each of the *n* stations of the fleet, including those that are already balanced (in order to check the state of the station and eventually trigger an intervention, is necessary). At the end of the tour, the trailer goes back at the warehouse, eventually with some bikes. Each station is visited exactly once. We must naturally respect the capacity of the trailer.

- **Data**. We know in advance :
  - The number $\boldsymbol{n}$ of stations visited during the tour and their geographical location : the stations are located in a coordinate system. In this coordinate system, one unit corresponds to an hectometer (1u=100m), inside a square of 10km side length. Concretely, we know the coordinates (x, y) of each station. The coordinates of the warehouse are also known ;
  - The capacity $\boldsymbol{K}$ of the trailer (it is the maximal number of bikes that can be carried in the trailer) ;
  - The distance $\boldsymbol{d_{ij}}$ between two stations *i* and *j* (and the distance between the warehouse and each station) is computed thanks to the coordinates of the different stations and of the warehouse. You will necessarily round the euclidean distance (with the function `round(Int64, x)`).
  - The number nbp_i of bikes present at the station *i* and the maximal capacity cap_i of the station *i*;

- The number *ideal$_i$* that represents the ideal number (not too many, not too few) of bikes that should be present at the station i for an optimal functioning of the station (this number is about 75% of the maximal capacity of the station, but it also depends on the location of the stations. Anyways, it is known).
- Remark : some bikes may be stolen : $\sum_{i=1}^{n} nbvp_i \leq \sum_{i=1}^{n} ideal_i$

- **Goals of the tour** : two questions arise. First we want to know wether it is possible or not to reset each station at its ideal level within a unique tour. We will say that a station has been balanced when it has been reset at its ideal number of bikes. If, after the tour, a station has not its ideal number of bikes, the number of bikes above or under the ideal number is called « imbalance of the station » (this number is nonnegative). **The first objective** consists of organizing the tour in order to minimize (and if possible to annul) the global imbalance, defined as the sum of the imbalances of each station. The second question is to know, among all the possible tours that minimize the global imbalance, which one minimizes the distance traveled by the trailer. A **second objective**, having a lower priority than the first one, consists of minimizing the total distance traveled, after having minimized the imbalance. So, an **optimal solution** of the problem will be a solution for which the value i^* of the global imbalance is minimal and for which the distance traveled is the smallest one among all the solutions having an imbalance equal to i^*.

- **Suggestions for the modeling**

Several formulations of this problem can be considered. We suggest a quadratic formulation with 0-1 and integer variables. The deadline for handing in the mathematical model is December 14th 2022, 23:59 (by email). Then, during the session of December 15th, a quadratic model and its linearization will be proposed : for the rest of the project, you will then be able to use your own formulation (sent on 12/14/2022) of the model proposed by the teacher on 12/15/2022.

We suggest to use two types of variables :

- **some 0-1 variables 0-1 x_{ij}** with the following meaning : x_{ij} equals 1 if and only if the station i is the jth visited station. The tour has n « steps » (a step corresponds to the visit of a station). The departure from the warehouse and the return to the warehouse are imposed and they are not considered as steps of the tour (but the distances traveled from the warehouse to the first visited station and from the last visited station to the warehouse must be counted in the total distance).

- **some integer variables** (nonnegative) ***load$_j$*** (j ranges from 0 to n) represent the number of bikes that are present on the trailer. The initial content of the trailer, at the departure of the warehouse, is *load$_0$*. The variable *load$_j$* (for $j \geq 1$) represents the number of bikes present on the trailer after the jth step.

- **some integer variables** (positive, negative or zero) ***drop$_{ij}$*** representing the number of bikes dropped at the station i (if some bikes are missing) or removed from the station i (if some bikes are in excess) at the step j. We must ensure that *drop$_{ij}$* can be different from 0 only if the station i is visited at setp j.

Important remarks :

- The total distance traveled by the trailer is intrisically quadratic if we use the variables $\boldsymbol{x_{ij}}$. We can linearize the function representing the distance in order to obtain a linear model using mixed variables (some of the variables are real, others are 0-1, others are integer).

- The global imbalance is a the sum of the individual imbalances. Each individual imbalance represents a notion of error. It is natural to represent this error with a norm. Here we use the absolute value of the difference with the ideal (we could have considered a quadratic error). Therefore we have to représentent an absolute value with a linear function and constraints.

- You will try to write a unique model allowing us to take into account the two objectives.

3) Instance format

The instances you have to solve can be find on moodle.

The basic example, illustrated in the next section is the instance mini_6 :

```
name mini_6
K 6
stations
# id x y nbp capa ideal
1 82  9  8 11 8
2 69 79  3  9 7
3 71  9 10 10 7
4 64 93  2  6 4
5 65 13  5  7 5
warehouse 50 50
```

File mini_6.dat

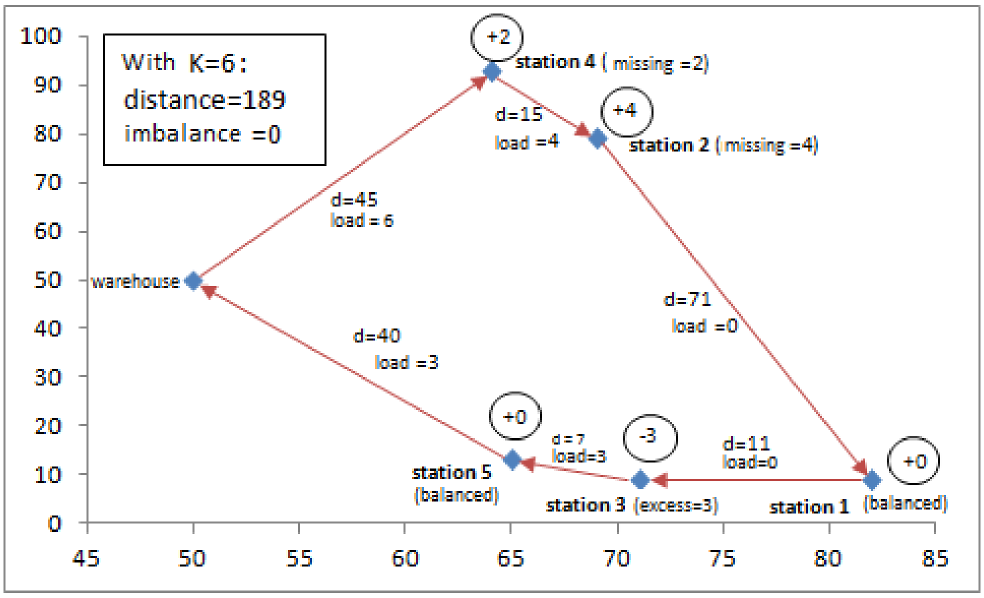
All the instance files respect the same syntactical rules. A line beginning with ‘#’ is a comment. The string after the keyword ‘`name`’ is the name of the instance. The number after the keyword ‘`K`’ is the capacity of the trailer. The lines after the keyword ‘`stations`’ and located before the keyword ‘warehouse’ correspond to the characteristics of each station : an identifier, its abscissa, its ordinate, its ‘nbp’ (number of bikes present before the tour), its ‘cap’ (capacity of the station), and its ‘ideal’ (ideal number of bikes). A last line contains the keyword ‘`warehouse`’ followed by the coordinates (x, y) of the warehouse.

4) Three examples and their solution

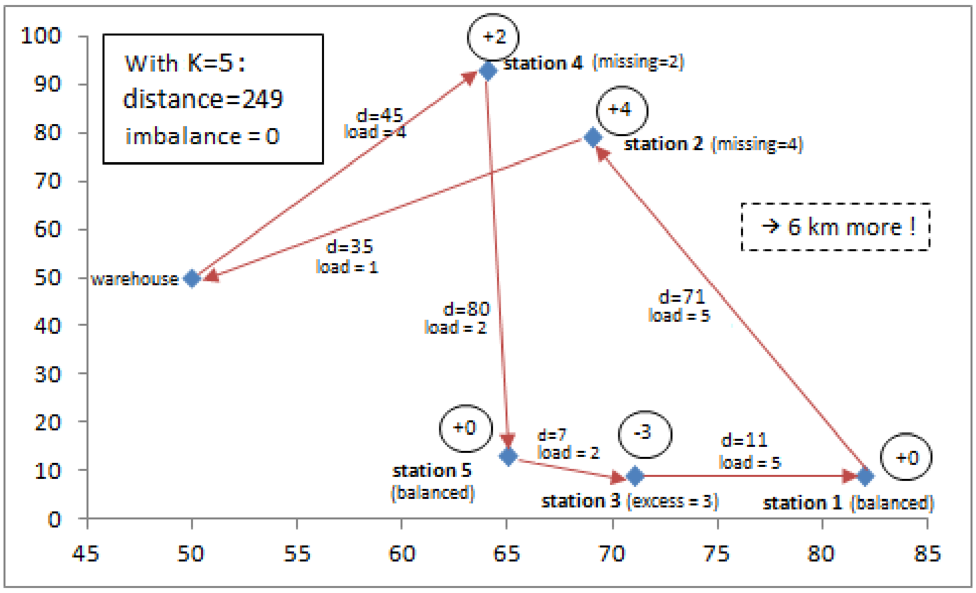
To illustrate the optimization work, we will consider 3 variants of the same instance, the instance mini_6 of the previous section, in which we change the value of the parameter *K* (capacity of the trailer). The instances mini_5 and mini_2 differ from the instance mini_6 only on the value of K (equal to 5 for mini_5 and 2 for mini_2). Solving the three instances yields the following optimal solutions.

Legend :

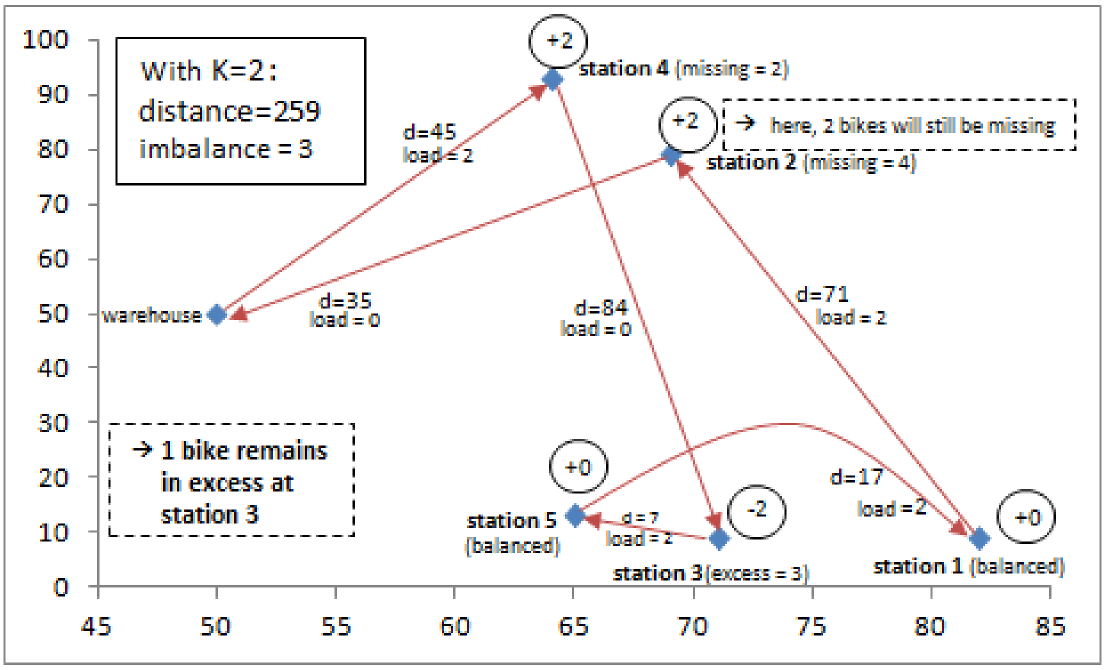
- The mention in brackets indicates the initial situation of the station
- ⓧ indicates the number of bikes dropped ($x \geq 0$) at the station or removed ($x \leq 0$) from the station
- The stock indicates the number of bikes present in the trailer
- The unit of distance is the hectometer (1 unit = 100 meters)



Optimal solution of the instance mini_6



Optimal solution of the instance mini_5



Optimal solution of the instance mini_2

5) Detailed work plan, output format and deadlines

: You have to

- **Propose a model** of the problem. You have to hand in your mathematical model before December 14th 23:59. You may propose a different formulation than the one that is suggested. In case of a quadratic model, its linearization is asked. A correction will be presented during the session of December 15th (therefore no delay can be accepted for the model).
- **Write** the (linear) model in Julia/JuMP and **solve** the instances with the solver Cbc. You may limit the exact solution by the solver to 5 minutes. If the optimal solution cannot be found within 5 minutes, you will have to indicate it in your final report.
- Determine a good solution of the problem, not necessarily the optimal solution, using a heuristic method or a meta-heuristic such as simulated annealing. You may invent an algorithm to find a solution of the problem, as good as possible (without using the solver). This will probably be the only way to solve the larger instances.
- Determine **lower bounds** of the imbalance. You will have to modify your model written for the exact solution in order to compute the continuous relaxation of the problem. The lower bound provided by the continuous relaxation will be usefull to evaluate the quality of your heuristic method.
- **Write a report** indicating the obtained results. You have to hand in your report not later than the day of the oral defense of your project (January 12th). This report must contain :
  - the code of your programs (in appendix)
  - an explanation of the idea of your heuristic method,
  - the ILP models,
  - a presentation of the numerical results obtained on the instances mentioned in section 3. For this presentation, complete as far as you can the following table. You

may eventually add some other informations if you think they are relevant. You may present the results in several tables for a greater readibility.

Instance			Exact solution					Heuristic			Lower bound of the imbalance	Output file
name	n	K	Optimum found within less than 5 min (yes/no)	CPU time (s) for the optimal solution	# nodes of the B&B	Imbalance	Distance	Imbalance	Distance	CPU time (s)		
mini_6	5	6	yes	6.84	0	0	189	0	189			mini_6.sol
mini_5	5	5	yes	7.38	0	0	249	0	249			mini_5.sol
mini_2	5	2	yes	1.82	0	3	259	0	259			mini_2.sol

The last column of the table indicates the name out the output file that you create (and join) when the optimal solution will be obtained in less than 5 minutes. This output file must respect the following syntax. Each line beginnig with a '#' will be a comment, then, in order, a line beginning with the keyword 'name' must indicate the name of the instance, a line beginning with '`imbalance`' will indicate the global imbalance, a line beginning with '`distance`' will indicate the total distance traveled, a line beginning with '`init_load`' will indicate the initial content of the trailer at the beginning of the tour, then a line will contain the keyword '`stations`'. After that, in the order of the optimal tour found, you will write a line per station visited, by indicating its identifier, the number of bikes dropped (positive value) or removed (negative value). The file ends with a line containing the keyword '`end`' :

```
name mini_2
imbalance 3
distance 259
init_load 2
stations
4 2
3 -2
5 0
1 0
2 2
end
```

file mini_2.sol