Discrete Optimisation
Project 2: Car on the Hill

10th November 2015

The Sart-Tilman university campus is crowded, especially its roads and car parks. To improve the situation, one solution would be to work on public transportation means or road infrastructure; on the other hand, this project follows another approach: make a better use of existing resources.

Current traffic jams are caused by an excessive number of cars trying to get to the campus; however, most of them contain only one person. To reduce the number of cars trying to get to the university, an obvious solution is to fill more the existing cars: carsharing. The effect might be two-fold: fewer cars will be used to get to the campus, but also fewer people will use the public transportation, making room for others and thus making the travel a more enjoyable experience for everyone.

1 Statement

This project is about statically optimising the use of cars. The carsharing application is centralised, and knows who has a car and the origin and destination for each user of the application (this data does not evolve in real-time). It will impose paths to users in order to minimise the total number of cars on the campus. The only degree of freedom for drivers is the maximum length of the path they agree on taking. The objective function is to improve the use of the existing cars, and thus to maximise the number of people getting to the campus by car (otherwise, the best solution would be to rely on public transportation only, with no cars on the campus).

Some users of the applications are tagged as drivers: they always take their car to go the to university. The driver cannot choose its path through the graph, but can impose a maximum distance. The path is chosen by the solver, with the guarantee that it has no cycles. Each car has a maximum number of passengers, which are assigned by the solver. The optimisation only takes place during the rush hour (timing constraints are not considered).

The area surrounding the campus is represented as a directed graph. All users start from some point and go to some other, following some path in the graph; the origin and the destination are not interchangeable. A passenger can only go in a path going from the origin of the driver to the destination, and not the other way around. However, it is not mandatory that they start from or end at the university campus.

2 Available data

The graph can be considered as a given. Each edge is associated to a length. A small graph will be made available in due time; however, you should test on a more realistic and much larger graph of the Liège area (with scopes such as communes, highways, main roads, secondary roads, etc.), extracted from databases such as Google Maps or OpenStreetMap.

Each person is described as an origin and a destination. Driver are persons for which more data are available: a maximum distance for their path (in kilometres) and a maximum number of passengers in their car (at least zero).

Other data can be derived from these and used in the model, including through non-obvious transformations, as long as this preprocessing does not perform the actual optimisation process.

3 Instructions

The project must be carried out by groups of two. It is divided in two parts.

1. The first one is about mixed-integer linear modelling. You are expected to hand in a **brief** report (PDF format, maximum six pages) on or before Sunday, 22nd November 2015 on the submission platform. It should contain a description of your complete model, with the variables, the objective function, and the constraints.

2. The second one is about the actual implementation. You are expected to hand in your code (ZIP or 7Z archive) on or before Sunday, 13th December 2015 on the submission platform, along with a short report detailing your progress since the model (maximum ten pages, included in the archive). It should contain an implementation of your MILP model and another way to solve the problem (ideally to optimality), without mathematical programming (for example, using graph algorithms such as the k shortest paths, or constraint programming); the two implementations may be intertwined (for example, using a heuristic solution to guide the optimisation process: good starting point, new solutions at MIP nodes improving current incumbent, generating cuts based on the heuristic solution, etc.). The choice of tooling is limited to the following list: Julia (0.3 or 0.4, please indicate the version you use) and JuMP; Java or C++ and Gurobi's API; callbacks can be used. Particular attention should be paid to *efficiency*.

The presentations will happen on Friday, 18th December 2015. Each group will have at most ten minutes to present their work, followed by questions. The main points should be the algorithm you developed, and how you made this algorithm and/or your mixed-integer linear model tractable.