High-Performance Implementation of Network Representation and Routing Algorithms

Problem Description
Mobility-on-demand services emerged a few years ago and their popularity increased significantly since then. In many US cities, Uber serves more demand than traditional taxi services. With the emergence of autonomous vehicles, their growth is only expected to grow even further. Ridepooling, i.e. the matching of multiple customers and their concurrent transportation by a single vehicle, will likely be necessary in order to build more efficient systems. However, the number of possible combinations to match customers grows exponential. In recent years, a methodology with graph-theoretic considerations was developed to match hundreds to thousands of requests at a time. These matching algorithms require a huge amount of routing queries to evaluate travel times between various network locations. However, to avoid dealing with additional computational times due to routing, models in literature typically assumed the availability of a travel time matrix for all nodes in the street network. For a real-life application, travel times are varying and need to be computed on demand. Highly efficient network representation and routing implementations (one-to-one, many-to-one, many-to-many) are necessary to determine system scales that can be treated in real-time.

The research group at the Chair for Traffic Engineering and Control has built a fleet simulation environment in python. This simulation module works sufficiently efficient as long as the network can be preprocessed and computation time is (nearly) independent of routing computations, but could benefit from a faster network and routing implementation for the case of dynamic travel times in larger networks.

Task Description
- Literature Review on currently available routing algorithms and their use-cases (1-to-1, many-to-one, many-to-many)
- Implement optimized network and routing module for python (i.e. C++ extension, cpython, numba, …) and evaluate performance for different use-cases
- Python code optimization:
  - Evaluate integrability of implemented routing engine within fleet simulation environment
  - Revise current implementation of parallelization
  - Optional: Observe, revise and optimize most time critical code segments in simulation framework
Contact:
Dipl.-Phys. Florian Dandl (florian.dandl@tum.de)
Roman Engelhardt, M.Sc. (roman.engelhardt@tum.de)
Lehrstuhl für Verkehrstechnik
Arcisstraße 21, 80333 München
Tel. +49 89 289 28594
www.vt.bgu.tum.de