Motivation: In smart cities, it is important to provide smart solutions for traffic management, especially with the growing population. Here we provide, develop and test new possibilities of traffic optimizations through sophisticated simulations.

Goal: Optimize the public transport routes of the city of Aarhus, Denmark and try to transport as many pedestrians as possible with buses in the shortest time possible without overloading the traffic.

Methods:

1. Extract the street network using SUMO (Simulation of Urban MObility) Software.
2. With the given API identify the pedestrians location and their destinations.
3. Generate buses which can fulfill the travel needs of the pedestrians.
4. Try to transport as many pedestrians as possible in the shortest time possible without overloading the traffic.
5. Optimize the public transport routes in various ways. **Optimization function**

    1. Level 1
        1. Maximum number of simultaneous buses/*total number of pedestrians in this round of simulation*
        2. Total number of passengers not transported at the end of the simulation/*total number of pedestrians in this round of simulation (ratio or percent)*
    2. Level 2
        1. Maximum number of simultaneous buses
        2. Total number of kilometers driven by all buses
        3. Total wait time of all transported passengers
        4. Total number of passengers not transported at the end of the simulation
    3. Level 3
        1. Maximum number of simultaneous buses of each type
        2. Total number of kilometers driven by all buses per type of bus
        3. Total waiting time of all transported passengers
        4. Total number of passengers not transported at the end of the simulation

Current state:

Our current solution aims at maximizing the number of transported passengers using L-type buses. The number of released buses is determined roughly by the total number of

passengers divided by the capacity of the bus type. This was a pragmatic choice given that we were not able to drop off passengers from buses (see “main problems” below). For every pedestrian appearing on the map, a new bus is released until the pre-defined number of buses is reached. Further pedestrians are then assigned to the buses based on a queue. The aim of this method aims to reduce the waiting time of the pedestrians.

The bus routes optimization can be simplified into a Capacitated Vehicle Routing Problem with Pickup and Delivery and with Time Window (CVRPPDTW), the goal is to find the optimal routes for multiple vehicles visiting a set of locations achieving either loading or unloading of a user (https://developers.google.com/optimization/routing/vrp#model) with multiple constraints. In one classical VRP problem, the start and end depot of the buses should be the same location. Although this prerequisite is not the case in the challenge, we can still use these available tools, since during the simulation, it has been found that the start and the end depot are very near to each other.

One example of such tools is the Google OR-Tools, an open source software for combinatorial optimization, which seeks to find the best solution to a problem out of a large set of possible solutions with potential additional temporal and capacity constraints (https://developers.google.com/optimization/introduction/overview). Our team tries to implement this tool to optimize the VRP problem.

An optimization problem usually consists of a set of target functions, and one or more constraint functions. In this challenge, the targets are, from level 1, to minimize the maximum number of simultaneously operating buses (in order to avoid traffic congestion) and to reduce the number of passengers left in the end.

Main problems:

1. Installing Sumo Software and the traci API was not straight forward and half of the group were not able to, especially on the Windows machines. Accordingly our progress was limited by the number of people able to code and having the software installed.
2. It was not possible for us to adjust a vehicle’s route more than once with vehicle.changeTarget or vehicle.setRoute because this would overwrite the vehicle’s entire route and the vehicle would then only run the last assigned route. We circumvented this by fixing the route from the very beginning by concatenating routes.
3. We did not manage to find a way to drop off people from the bus at their intended destination, even with the nightly build. The pedestrians’ destination did not change. The destination edge coincided with the bus’ location edge and still the pedestrians

would not hop off. This constraint made it impossible to optimize the traffic because once a bus was full it became inoperative.

4. The optimization algorithm needs a proper connection with the main code, which takes the randomly generated user position as well as the time window as the input parameters then outputs the optimized routes for each bus and the corresponding users loaded by the specific bus.

Limitations:

- Our solution comes with the problem that some of the pedestrians might not be picked up at all because the timing is not correct: some of the buses might pass bus stops before the corresponding passenger appears.
- In the Google OR-Tools, transportation time between locations needs to be defined, which is dependent on the road network and speed of buses, additional assumptions including the ‘manhattan distance’ and a certain speed of buses instead of the maximal speed should be made.