Autonomous Scooter-On-Demand Sharing System Formulation

MATLAB Model structure:

Some parameters:
- vehicle_availability_map (position of each vehicle in the road network and its states (available/occupied/booked/down),the time step it's at (four dimensional)
- road represented as nodes every 10 meters along the road;

efloater_model.m
- main model, simulates the system for 1 month

origin_target_generation.m
- Customer arrival follows poisson process with arrival rate of lambda per unit time, where lambda can be location_variant, time_variant based on survey data (generate trip origin nodes)
- Sample from trip distribution to determine the trip target node for these origin nodes (the trip distribution table denoting the probability to travel between each origin-target pair can be estimated from survey data)

vehicle_location_initialization.m
- initialization of each scooter's location at the beginning of each day based on vehicle_availability_map at the end of last day and day of week, the rebalancing could be performed during 3-4 am

vehicle_assignment.m
- matches demand to the nearest "available" efloater and provide an estimated waiting time to arrive, if waiting time is within the customers' tolerance time, the status becomes "booked" and the customer waits at his position for the scooter to come, output waiting_time; if there is more demand than available vehicle, for demand that are not matched, reiterate till exceeds tolerance time,waited_time =+1.

- customer tolerance time could follow a distribution (based on survey data), then sample from the distribution

- At each time iteration, a "booked" scooter has a probability to change its status to "available" as a customer may cancel the order while waiting for the scooter to arrive; if not cancel but leave, change from "booked" to "available" upon arrival

Output:
- waiting_time (the time it takes to wait for the scooter to arrive when you've already got a scooter matched to you and is on its way)

- waited_time: counter to remember how much time customer has waited for an available scooter when there is no available scooter yet; if waited_time exceeds tolerance time, the person leaves the queue update vehicle_availability_map after each iteration, update the status
of each vehicle and its position on the map

(Assumptions: person stays at the same place while waiting for the
scooter to arrive)

total_time_calculation.m
- Calculate time to travel from the start_node (generated demand
location) to target_node (destination), not including waiting time at start node; output an updated vehicle availability map when the trip ends at current_time+total_time

fleet_rebalancing.m
- We divide the entire study area into zones, we then compute a balance value for each zone based on normalized available vehicle and expected demand. for zones with balance value exceeding the pre-set threshold value, we push/pull other available vehicle to/from the neighboring blocks. A virtual demand is generated at the center node of the block we want to rebalance and the vehicle nearest to that node is assigned for this demand, however, the vehicle's state still remains available to pick up other passengers along the way and the second nearest vehicle would be dispatched in this case and so on …

find_closest_vehicle.m
- find the node with the nearest scooter(s) that is available "A", note that
there might be many available scooters at the closest_vehicle_position
node

generate_manhattan_heuristic.m
- stores the distance to a targeted coordinate as heuristic (manhattan distance)

Modules to be included in the future:
- battery charging constraints (location of charging stations)
- scooter battery level constraints

Other fleet rebalancing strategies to be considered in the future:
Generate virtual passengers at each node (as opposed to real passengers which correspond to real demand), the variables to decide in this case would be arrival rate of virtual passengers (which is equivalent to rebalancing rate as virtual passengers can be seen as fictitious demand used for rebalancing) and the routing policy of the virtual passengers (rebalancing routing policy, which path to travel); The objective function would be to minimize the number of rebalancing vehicles on the road and to make the utilization of each node equal) (Source: Control of Robotic Mobility-on-Demand)