Autonomous Vehicles Impact on Ride-hailing: A Big-Data Analysis Project

1. Background

Mobile agents search for stationary resources in a road network. Each resource can be obtained by only one agent at a time, therefore, the search is competitive. A search problem of this nature arises in applications that are very commonplace in typical urban transportation systems such as the following:

- Crowdsourced taxicabs (mobile agents) looking for customers (stationary resources).
- Crowdsourced delivery vehicles looking for customers/packages.
- Vehicles (mobile agents) looking for available parking slots (stationary resources).
- Electric cars (mobile agents) looking for available charging stations (stationary resources).

Objective:

To determine how the assignment of resources to agents affects the performance of the system i.e. wait time of resources and empty cruise of agents. The assignment is fair for crowdsourced vehicles, and optimum for (owned) autonomous vehicles.

2. Problem Definition

The problem is defined in the context of crowdsourced and owned taxicabs searching for customers to pick up. The system consists of four types of entities: a road network, mobile agents (i.e., taxicabs), and stationary resources (i.e., customers), and an assignment authority. Agents are introduced to the system at once at the beginning of the operation, each located at a random location on the road network. The set of agents is fixed throughout the operation, the size of which will be referred to as the *agent cardinality*. Resources are introduced to the system in a streaming fashion, each with an origin and a destination. Each resource has a maximum life time (MLT) starting from its introduction, beyond which the resource will be automatically removed from the system, an outcome which we will call resource expiration. After an gent is introduced to the system, it is labeled as empty/unoccupied, and cruises along a path called a search path1. Every time period of x seconds, called the assignment period, the resources introduced during the period are matched with the unoccupied agents according to an optimal or a fair assignment.

Once an agent is assigned to a resource, the agent is labeled as *occupied* and the resource is removed from the system. Then the agent moves to the resource (for pick-up) and then to the destination of the resource (for drop-off), along shortest-travel-time paths. Once the agent arrives at the destination, it is labeled as empty. The agent knows neither when and where the future resources will be introduced, nor any information regarding other agents.

The project looks at the following three aspects of performance, in order of priority:

1. The *wait time* of a resource is the period of time from its introduction until its pick-up or expiration.
2. The *expiration percentage* is the percentage of expired resources.
3. The *search time* of an agent is the amount of time from when the agent is labeled empty until it picks up a resource. An agent experiences multiple search times, each corresponding to one

[1] Examples of a search paths: 1. choose a random location and go there, 2. random walk, 3. go to closest transportation hub or taxi station, remain at the drop-off location.

assignment.

Output:

Each performance criteria as a function of the size of the assignment period, for the fair (crowdsourced) and the optimal (owned av's) assignments. Each of the above plots may depend on the agent search strategy (random walk, closest taxi station), as well as the way of measuring the benefit (e.g. minimum customer wait time, maximum benefit)

3. Software Framework

There are multiple constraints that need to be satisfied by a submission:

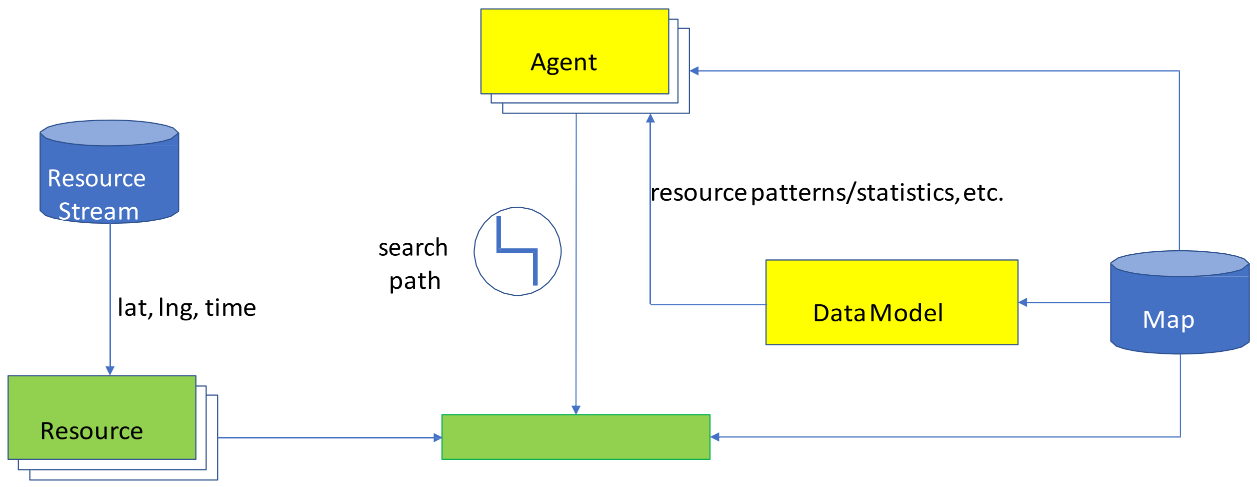
1. An agent does not know when and where resources will be introduced in the future.

2. The agents do not share information with each other;

3. The travel paths of an agent are constrained to the road network and adheres to traffic limitations provided by the map (e.g., route restrictions, speed limits).

To ensure that these constraints are satisfied, the project can use an open-source simulation framework called COMpetitive SEarching Testbed (COMSET). COMSET, written in Java, implements the fundamental logic of the system described in Section 2, including the handling of maps, the injection of resources, the movement of agents along a given path, the assignment of agent to resource, etc. COMSET comes up with two search path strategies: 1. choose a random location and go there, 2. random walk. A team may also implement their own search path strategies. A reasonable candidate is going to the closest transportation hub or taxi station, and waiting there. The architecture of COMSET is shown in Figure 1.

Figure 1: The architecture of COMSET



The major software modules in COMSET are:

- Agent: computes search path based on the map and optionally the result of the data analyzer.
- Assignment Authority: tracks the current position and status of the agents; assigns agents to their nearest resources; computes the search times, wait times, and expiration percentage.

Details of COMSET are provided on Github (https://github.com/Chessnl/COMSET-GISCUP/)

Dataset

Manhattan dataset which is a subset of the New York TLC Trip Record YELLOW Data with only the records such that both pick-up and drop-off are within the Manhattan area. The agent cardinality can be set in a range of 5000-10000.