Scram Applied to Car Sharing

Abstract

Introduction

We seek to improve the performance of a fleet of shared autonomous vehicles through improved matching of vehicles to passengers requesting rides.

Car Sharing Model

For experiments we use the agent-based car sharing model proposed by Fagnant and Kockelman (Fagnant and Kockelman 2014). This model represents a 10 mile by 10 mile city as a grid of 0.25 mile by 0.25 mile grid cells. Vehicles can move freely within cells and can move up, down, left, or right to adjacent cells. Diagonal motion is not permitted and cars do not interfere with each other's path. Time is discretized into five minute intervals for a total of 288 time steps per day. Each run of the simulation corresponds to one 24 hour day.

Trips are generated in each grid cell according to a rate that decreases the farther a cell is from the city center. This rate is the mean for a Poisson process from which a number of trips is drawn for the corresponding cell. The distance and start time of each trip is drawn from a distribution based on U.S. National Highway Traffic Safety Administration trip distance data. While all trips for the day are generated at the start of the run requests for rides are not made until the start time of the trip.

At each time step available vehicles are matched to passengers requesting trips. If a trip cannot be served it is added to a wait list and can request a ride again at the next time step. Vehicles that could not serve trips because they were low on fuel go to the nearest fueling station (assumed to be within the same grid cell) and are out of service for two time steps (10 minutes). Then vehicles move to pickup passengers. Vehicles already with passengers proceed to their trip destination. Each vehicle can move a fixed number of grid cells depending on the current maximum speed.

We consider two schemes for determining the number of cars that are needed to provide adequate service. As in (Fagnant and Kockelman 2014), the number of cars can be determined by running the simulation for 20 runs with each run initialized with zero cars. The simulation cycle proceeds normally except after a trip has waited for 10 minutes a new car is generated at the start location of the trip. The number of cars generated is an estimate of the required fleet size. The average number of cars generated across these 20 runs is the fleet size. A final run repeats this procedure until the correct fleet size has been generated. Alternatively, since we wish to compare the sensitivity of different methods to variations in the fleet size, we can specify a fixed number of vehicles. In this case the above procedure can be repeated until the requested fleet size is reached and then no more cars are generated.

This model approximates the effects of a car sharing system in a 10 mile by 10 mile urban area. Car speed has small variations due to time of day and distance from the city center but is otherwise fairly constant. The simulation only models vehicles and trips using the car sharing service. The presence of other modes of transportation is reflected in the trip generation rates and speed variations but these are only approximations. Nonetheless the model has been used in other car sharing studies (Fagnant, Kockelman, and Bansal 2015) to date and is realistic enough for the purpose of this work.

Matching Algorithms

Decentralized Greedy Matching Without a centralized trip-car assignment system trips would need to request the nearest vehicle similar to how users of Uber or Lyft look for nearby vehicles. This decentralized method can be implemented as a greedy matching of trips to vehicles. In a random order, trips are assigned to the closest vehicle to them. If one is not available they are added to a wait list.

Centralized Greedy Matching Each trip looks for a car within a one cell radius of itself. After all trips have made this search the radius is increased and trips search again. This radius can either increase up to searching the entire city or terminate at a fixed limit (e.g. the distance a vehicle can travel in one time step).

Hungarian Algorithm A greedy approach to the assignment problem is likely to be suboptimal. A natural improvement is the Hungarian algorithm (Kuhn 1955) which finds a minimum cost perfect matching in a bipartite graph.

Minimal Makespan Matching The Hungarian algorithm produces a minimum cost perfect matching but the optimal solution does not consider the makespan problem. In our scenario this equates to the possibility that some trips have a much longer wait time even though the total wait across all trips is minimized. To improve on this we use a minimal-makespan matching algorithm to find an assignment that minimizes the longest distance that any vehicle must travel to a passenger. Specifically we implement the Minimum Maximal Distance + Minimum Sum Distance^2 (MMD+MSD^2) proposed in (MacAlpine, Price, and Stone 2015).

Experimental Setup

To compare different approaches we run two sets of experiments. The first set of experiments compares the performance of the above algorithms in trials of 100 consecutive days. In these experiments the fleet size is constant to observe the strengths of the different algorithms under the same settings.

Results

References

Fagnant, D., and Kockelman, K. 2014. The travel and environmental implications of shared autonomous vehicles, using agent-based model scenarios. *Transportation Research Board Part C* 40:1–13.

Fagnant, D.; Kockelman, K.; and Bansal, P. 2015. Operations of a shared autonomous vehicle fleet for the austin, texas market. *Transportation Research Board*.

Kuhn, H. W. 1955. The hungarian method for the assignment problem. *Naval research logistics quarterly* 2(1-2):83–97.

MacAlpine, P.; Price, E.; and Stone, P. 2015. Scram: Scalable collision-avoiding role assignment with minimal-makespan for formational positioning. In *AAAI Conference on Artificial Intelligence (AAAI)*.