Predictive positioning and quality of service ridesharing for campus mobility on demand systems

The MIT Faculty has made this article openly available. Please share how this access benefits you. Your story matters.

<table>
<thead>
<tr>
<th>Citation</th>
<th>Miller, Justin, and Jonathan P. How. “Predictive Positioning and Quality of Service Ridesharing for Campus Mobility on Demand Systems.” 2017 IEEE International Conference on Robotics and Automation (ICRA), May 2017, Singapore, Singapore, 2017.</th>
</tr>
</thead>
<tbody>
<tr>
<td>As Published</td>
<td><a href="http://dx.doi.org/10.1109/ICRA.2017.7989167">http://dx.doi.org/10.1109/ICRA.2017.7989167</a></td>
</tr>
<tr>
<td>Publisher</td>
<td>Institute of Electrical and Electronics Engineers (IEEE)</td>
</tr>
<tr>
<td>Version</td>
<td>Original manuscript</td>
</tr>
<tr>
<td>Accessed</td>
<td>Thu Jan 03 17:07:38 EST 2019</td>
</tr>
<tr>
<td>Citable Link</td>
<td><a href="http://hdl.handle.net/1721.1/114725">http://hdl.handle.net/1721.1/114725</a></td>
</tr>
<tr>
<td>Terms of Use</td>
<td>Creative Commons Attribution-Noncommercial-Share Alike</td>
</tr>
<tr>
<td>Detailed Terms</td>
<td><a href="http://creativecommons.org/licenses/by-nc-sa/4.0/">http://creativecommons.org/licenses/by-nc-sa/4.0/</a></td>
</tr>
</tbody>
</table>
Predictive Positioning and Quality Of Service Ridesharing for Campus Mobility On Demand Systems

Justin Miller and Jonathan P. How

Abstract—Autonomous Mobility On Demand (MOD) systems can utilize fleet management strategies in order to provide a high customer quality of service (QoS). Previous works on autonomous MOD systems have developed methods for rebalancing single capacity vehicles, where QoS is maintained through large fleet sizing. This work focuses on MOD systems utilizing a small number of vehicles, such as those found on a campus, where additional vehicles cannot be introduced as demand for rides increases. A predictive positioning method is presented for improving customer QoS by identifying key locations to position the fleet in order to minimize expected customer wait time. Ridesharing is introduced as a means for improving customer QoS as arrival rates increase. However, with ridesharing perceived QoS is dependent on an often unknown customer preference. To address this challenge, a customer ratings model, which learns customer preference from a 5-star rating, is developed and incorporated directly into a ridesharing algorithm. The predictive positioning and ridesharing methods are applied to simulation of a real-world campus MOD system. A combined predictive positioning and ridesharing approach is shown to reduce customer service times by up to 29%, and the customer ratings model is shown to provide the best overall MOD fleet management performance over a range of customer preferences.

I. INTRODUCTION

Mobility On Demand (MOD) systems have the potential to revolutionize transportation systems in urban settings by providing commuters access to vehicles without requiring private ownership. In such systems, a fleet of shared vehicles continually services multiple customers by transporting them from their requested on demand pickup location to their desired destination. It is estimated that by 2030, as much as 26% of all global miles traveled will be from customers using shared vehicles [1]. A fundamental challenge for MOD systems is providing a high customer quality of service (QoS) in order to minimize any drawbacks that customers may experience by relying on the shared resources.

There are many factors that can affect a customer’s QoS such as cost, comfort, safety and convenience [2], many of which can be improved through the use of autonomous MOD systems composed of self-driving vehicles [3]. Several autonomous MOD fleet management strategies have been introduced as a means of improving QoS. These approaches attempt to find a “rebalancing” policy that redistributes vehicles within the system based on customer demand using either a fluid model approach [4], a Markov transition model approach [5], a queueing-theoretical model approach [6], or a model predictive control approach [7]. The approaches were developed to operate on city-wide scales and assume that an appropriately large fleet of autonomous vehicles is available, an assumption that does not yet reflect the current state of real-world autonomous MOD systems. In practice, autonomous MOD systems are being deployed with limited sized fleets [8]–[10]. This work is further motivated by our own MOD system operating on MIT campus, where autonomous fleet management strategies are applied to a fleet of only three human-driven vehicles, shown in Figure 1a. Previous rebalancing methods do not scale well when applied to such small fleets. For example, the policies are designed to redistribute multiple excess vehicles to relatively few network arrival locations. The methods breakdown when the reverse is true and fleet sizes are relatively small compared to the number of network locations such that there generally aren’t enough vehicles to cover all arrival locations. This work seeks to address this challenge by instead identifying key locations in the MOD system which minimize expected customer wait times regardless of fleet size.

Additionally, those previous works only consider the use of single capacity vehicles and do not address the benefits and challenges of utilizing ridesharing. In ridesharing, multiple customers may share a ride in an MOD vehicle at the same time. Newly arrived passengers can be picked up before onboard passengers have been dropped off, allowing for more customers to be serviced with fewer vehicles. However, the reduced wait time for requested passengers comes at the expense of increased ride time of onboard passengers. With ridesharing, perceived QoS becomes dependent on customer preference (i.e. how much customers prefer one service metric over another). A large body of work has studied
a form of ridesharing known as the Dial-A-Ride Problem (DARP), which is a specialization of the Vehicle Routing Problem formulated specifically for the transportation of customers. DARP problem formulations typically take either the form of an integer program or a scheduling problem. Integer program formulations encode each customer QoS metric as a decision variable and use heuristic methods such as genetic algorithms [11], simulated annealing [12], and tabu search [13] to minimize an objective, such as a cost function composed of a weighted sum of the metrics. Scheduling problem formulations enumerate the possible ways of inserting new passengers into vehicle schedules, and encode customer QoS metrics as feasibility constraints [14]–[16]. The main challenge with all of these approaches is that the weights or constraint thresholds that define customer preference may be chosen incorrectly and even the structure that encodes the QoS metrics could be wrong. In general determining customer preference can be difficult. However, many MOD systems such as Uber, Lyft, and the MIT MOD system are able to query passengers for feedback in the form of a 5-star rating using a ride request app, as shown in Figure 1b. This work takes advantage of this available information through a ridesharing algorithm which does not encode customer preference directly, but rather utilizes a learned customer ratings model when solving the DARP scheduling problem. The contributions of this work are: 1) a predictive positioning approach that minimizes expected customer wait time in MOD networks with fewer vehicles than customer arrival locations; 2) a customer rating model which learns from 5-star rating feedback and serves as a customer QoS-focused cost function; and 3) a schedule-based ridesharing framework that accommodates customer QoS metrics without the need for encoding customer preference constraints.

II. PROBLEM FORMULATION

1) Customer Arrival Model: An MOD network is modeled as a directed network graph denoted by \( \mathcal{G} = (\mathcal{V}, \mathcal{L}) \), where \( \mathcal{V} = \{v_1, \ldots, v_N\} \) is a set of \( N \) nodes, and \( \mathcal{L} = \{l_1, \ldots, l_{N_L}\} \) is a set of \( N_L \) directed link edges each taking the form of an ordered pair of neighbor nodes \( l = (n_i, n_j) \in \mathcal{N}^2 \). A route \( r(n_i, n_j) \) is defined as a sequence of directed links \( \mathcal{L}_r \subseteq \mathcal{L} \) which corresponds to a unique minimum-travel-time path between a pair of nodes \( (n_i, n_j) \). Customers are assumed to arrive at nodes in the network graph according to a Poisson process with arrival rate parameter \( \lambda_n \). In many MOD systems, customer arrival rates will be time-varying and may feature large fluctuations on short-time scales. In this work, a discrete-time approximation is used where Poisson arrival rates are static for the operating duration, with short-term fluctuations averaged out across the time period.

2) Ridesharing: Upon arrival, customers send ride requests consisting of a pickup node \( p \in \mathcal{V} \) and a drop off node \( e \in \mathcal{V} \). Let \( \mathcal{C} \) be the set of customers who have requested rides, let \( \mathcal{O} \) be the set of customers, let \( \mathcal{C} = \{c_1, \ldots, c_N\} \) be the set of requested customer pickup and drop off nodes, and let \( \mathcal{C}_v \) be the set of vehicles in the MOD fleet. Vehicle \( v \) will service a subset of the customers \( \mathcal{C}_v \subseteq \mathcal{C} \) by visiting their pickup and drop off nodes. All customer nodes are inserted into a schedule \( s_v = \{s_1, \ldots, s_{N_v}\} \), where the vehicle traverses in order using a sequence of routes. Let \( Q \) be the maximum capacity of each vehicle. The ridesharing problem is that of finding both the customer to vehicle assignments as well as the vehicle schedules such that the cost of scheduling all customers is minimized. A four-index ILP formulation for the DARP is proposed as an extension of the models presented in [17], with emphasis placed on the ordering of customers within schedules. The decision variables \( x_{cij}^v \in \{0,1\} \) are equal to 1 if vehicle \( v \) is assigned customer \( c \), with \( p_c \) and \( e_c \) respectively sequenced at \( s_i \) and \( s_j \); and zero otherwise. The objective of the ridesharing problem formulation is to minimize the total customer QoS cost, that is,

\[
\begin{align}
\min & \quad \sum_{v \in \mathcal{V}} \sum_{c \in \mathcal{C}} \sum_{i=1}^{N_v} \sum_{j=1}^{N_v} g_{cij}^v x_{cij} \\
\text{s.t.} & \quad \sum_{v \in \mathcal{V}} \sum_{i=1}^{N_v} \sum_{j=1}^{N_v} x_{cij}^v = 1 \quad \forall c \in \mathcal{C} \\
& \quad \sum_{i=1}^{N_v} \sum_{j=1}^{N_v} x_{cij}^v = 0 \quad \forall v \in \mathcal{V}, c \in \mathcal{C} \\
& \quad x_{cij}^v = 1 \quad \forall v \in \mathcal{V}, i,j \in \{1, \ldots, N_v\} \\
& \quad \sum_{c \in \mathcal{C}} \sum_{i=1}^{N_v} \sum_{j=k+1}^{N_v} x_{cij}^v \leq Q \quad \forall v \in \mathcal{V}, k \in \{1, \ldots, N_v\} \\
& \quad x_{cij}^v \in \{0,1\} \quad \forall v \in \mathcal{V}, c \in \mathcal{C}, i,j \in \{1, \ldots, N_v\},
\end{align}
\]
Algorithm 1: Predictive Positioning

1. **Input:** customer node arrival rates \( \{\lambda_1, \ldots, \lambda_{N_a}\} \)
2. **Output:** vehicle locations \( k^* \)
3. enumerate vehicle placement options \( K \)
4. enumerate possible arrival locations \( A \)
5. **for** \( a \in A \) **do**
6.   **for** \( k \in K \) **do**
7.     \[ w_{k,a} \leftarrow \text{computeTotalWaitTime}(k, a) \]
8.     \[ p_a \leftarrow \text{computeProbability}(a) \]
9. \( k^* = \arg\min_{k \in K} \sum_{a \in A} w_{k,a} p_a \)
10. **return** \( k^* \)

III. APPROACH

This section presents two new approaches for utilizing and solving the ridesharing problem, (1) to (5), in order to improve customer service under different operating regimes. The first approach manages vehicles in the absence of ride requests by using a predictive positioning algorithm, while the second approach manages vehicles to accommodate ride requests using a ridesharing algorithm. Additionally, two QoS focused cost functions are presented, a traditional weighted excess walk and solving the ridesharing problem, (1) to (5), in order to improve customer service under different operating regimes.

A. Predictive Positioning

The predictive positioning algorithm uses known customer arrival rates \( \{\lambda_1, \ldots, \lambda_{N_a}\} \) to find the key predictive nodes within the network graph to place unassigned vehicles such that the expected wait time for arriving customers is minimized. A special form of the ridesharing problem, (1) to (5), is solved for a sequence of \( N_a \) predicted customer arrivals, where \( 1 \) vehicles are already assigned to at most one predicted customer \( (Q = 1) \); 2) the number of considered customers is set to be the number of vehicles \( (N_a = N_s) \); and 3) the cost is set to be the customer wait time. Let the vector \( k \in \{0, 1, \ldots, N_a\}^{N_a} \) denote the number of vehicles located at each node and \( a \in \{0, 1, \ldots, N_a\}^{N_a} \) be the number of customer arrivals on each node. \( K = \{k \mid \sum_{i=1}^{N_a} k_i = N_a\} \) is the set of all possible vehicle placement options, and \( \mathcal{A} = \{a \mid \sum_{i=1}^{N_a} a_i = N_a\} \) is the set of all combinations of possible arrivals. The predictive node locations \( k^* \) for which to place vehicles is determined using Algorithm 1.

The wait time cost \( w_{k,a} \) in Line 7 is determined by using a greedy solution to the ridesharing problem. The wait times are assumed to be dependent only on the structure of the network graph, and therefore are computed and stored offline. To handle the cases where some vehicles are serving customers while others need to be predictively positioned, wait times are computed for all numbers of free vehicles from 1 to \( N_a \).

The probability of a set of arrivals \( p_a \) in Line 8 is determined using decomposition of the total network arrival Poisson process. Given that an arrival occurs, the probability of that arrival occurring at a node \( n_i \) is given by \( P(a_i = 1 \mid N_a = 1) = \frac{\lambda a_i}{A} \). The probability of a set of arrivals occurring according to \( a \) follows a multinomial distribution,

\[
P(a) = \frac{N_a!}{a_1! \cdots a_{N_a}!} \left( \frac{\lambda_1}{A} \right)^{a_1} \cdots \left( \frac{\lambda_{N_a}}{A} \right)^{a_{N_a}}.
\]

Arrival probabilities are computed online whenever customer arrival rates change.

B. Ridesharing

1) Ridesharing Algorithm: The ridesharing algorithm uses an insertion method as a heuristic solution to the ridesharing problem. Algorithm 2 presents the method for assigning a new customer request to a vehicle such that the total QoS cost to the system is minimized. The algorithm is executed online whenever a new customer ride request is submitted. Line 4 temporarily assigns the new customer to each vehicle generating a temporary customer allocation \( \hat{C}_v \). Line 5 enumerates all feasible ways of inserting the request into the schedule, where \( \hat{S}_v \) is the set of all feasible schedules \( \hat{s}_v \), and where feasibility is met by ensuring that \( \hat{p}_v \) is inserted before \( e_c \) and that vehicle capacity is not exceeded. Lines 6 and 7 find the best feasible schedule \( \hat{s}_v^\ast \) and corresponding new customer QoS metrics \( \hat{M}_v^\ast \) for each vehicle, where \( \hat{M}_v^\ast = \{\hat{m}_c \mid \hat{c} \in \hat{C}_v \} \). For comparison, Line 8 computes the original customer QoS metrics \( M_v \) for each vehicle, where \( M_v = \{m_c \mid c \in \hat{C}_v \} \).

Rather than imposing constraints on customer QoS metrics, the algorithm uses a virtual “rejection vehicle”, \( v = N_v + 1 \) to make bids that consider the case where the customer is not serviced by the MOD system. Line 9 computes the rejection vehicle customer metrics \( M_{N_v+1} = \{\hat{m}_{\text{rejected}} \} \). While seemingly counter-intuitive, rejections are in fact important for improving customer QoS. For example, if a customer’s wait time is significantly longer than the time it would take for them to walk, then they may prefer to be rejected rather than to wait to use the service. In this work, rejected customers are prevented from making additional requests, although this
could be adapted to allow customers to resubmit with QoS relaxations.

Line 10 finds the combination of baseline and new metrics that includes all customers and has the lowest overall customer QoS cost, and then returns the vehicle \( v^+ \) that contains \( c \). If \( v^+ \) is the rejection vehicle, then the customer is rejected, otherwise the schedule for \( v^+ \) is updated to accommodate the request.

The primary benefit of the ridesharing algorithm is the ability to evaluate customer QoS without having to encode the customer preference structure into the algorithm. For example, other methods [14]–[16] encode feasibility constraints on customer metrics such as wait time or ride time, where customers are rejected if these are not met. Instead, a more general approach is taken in Algorithm 2 where a competing bid is made to reject a customer, and the rejection is made only when the overall QoS of the system would be improved by doing so. This approach opens the door for a ratings based cost function where bids are made without constraining the customer metrics directly.

2) Ridesharing Cost Functions: Two ridesharing cost functions are presented which evaluate the customer QoS cost from a set of customer ride metrics. First, a cost function composed of a linear weighted combination of the customer metrics is proposed as

\[
g(m_c) = \sum_{i=2}^{m_c} w_i^{rej} t_c^{rej} m_{c,i} + w_i^{acpt} (1 - t_c^{rej}) m_{c,i},
\]

where \( w_i^{rej} \) and \( w_i^{acpt} \) are weights for metric \( i \) that are used to allow for differentiating between rejected and serviced customers. For example, a service time focused cost function would be \( g(m_c) = t_c^{rej} t_c^{walk} + (1 - t_c^{rej}) t_c^{service} \) where the cost is the service time if the customer receives a ride and the walk time if they are rejected. This form of cost function requires that the weights be properly chosen to reflect customer preference, and can result in poor customer QoS if the weights are wrongly chosen.

To overcome the need to choose weights, a second ratings based cost function is presented which learns and uses customer preference through feedback from 5-star ratings. The rating model utilizes a random forest of classification decision trees, based on the work of [18]. Random forest algorithms tend to prevent overfitting and have been demonstrated to perform well empirically [19]. To train the random forest model, a dataset \( D \) from the MOD system is collected in the form of 5-star ratings \( Y_{train} = \{y_1, \ldots, y_{N_D}\} \) and a ride metrics feature vector \( X_{train} = \{m_1, \ldots, m_{N_D}\} \) such that \( Y_{train} = RF(X_{train}) \) where \( RF(X) \) is the trained random forest. The trained random forest then serves as the ridesharing cost function such that \( g(m_c) = -RF(m_c) \) where the minus sign is included to maximize customer rating. It is assumed that a customer’s 5-star rating is given purely to reflect their ride metrics and not factors such as driver interactions which are not necessary for autonomous MOD systems.

IV. EXPERIMENTS

The predictive positioning and ridesharing methods are tested using simulation of the MIT MOD system. Specially, there are two motivating test cases: 1) evaluating how service times for customers are affected under different fleet management strategies as customer arrival rates grow; and 2) evaluating how customer QoS is affected by various ridesharing strategies operating under a range of unknown customer preference models. The MIT MOD system is used to provide simulation parameters that reflect a realistic operating environment for vehicles and customers.

A. Simulation Setup

Pedestrians and vehicles operate within a network graph for the MIT campus. The network graph, shown in Figure 2, is generated using pedestrian trajectory data collected from sensors onboard the MOD vehicles following the method presented in [20]. A two hour time period is simulated; during which time a subset of 10 randomly chosen nodes are assigned non-zero pedestrian arrival rates in order to reflect that not every campus location experiences arrivals at all times. Customer arrival rates are static for the time period and take values between 0 and 0.45 ped/min at each node. There are 3 vehicles in the MIT MOD system each with a maximum capacity of 3 passengers. Vehicles travel between nodes according to the schedule generated by the ridesharing algorithm. A vehicle picks up its assigned customers upon reaching a scheduled node. If a vehicle’s schedule is empty, the vehicle will travel to nodes prescribed by the predictive algorithm. A vehicle picks up its assigned customers upon reaching a scheduled node. If a vehicle’s schedule is empty, the vehicle will travel to nodes prescribed by the predictive positioning algorithm. Vehicle link speeds are either 11m/s for links corresponding city streets or 4m/s for campus pathways.

Predictive Positioning

The predictive positioning algorithm finds key predictive node positions to place unallocated MOD vehicles based on customer arrival rate parameters. Figure 3 shows the predictive nodes chosen for either 1, 2, or 3 unallocated vehicles for a particular set of arrival rates. The predictive nodes take into account the need to place vehicles in areas where customers are expected to arrive.

Fig. 2: Pedestrian traffic network on MIT campus with overlaid pedestrian trajectories. The network graph is composed of 27 nodes, 106 directed links, and 1056 precomputed routes.
account both the probability of the arrivals occurring and
the vehicles’ travel time to reach each node. If only a single
vehicle is unallocated, it will tend to be positioned centrally
within the network, but skewed towards large arrival rates.
When more vehicles are unallocated, the predictive nodes are
further spread out for better coverage.

The performance of the predictive positioning algorithm is
evaluated through comparison against a baseline unmanaged
MOD strategy, where vehicles respond to pickup requests but
are not repositioned after dropping off customers. The two
methods are first compared without the use of ridesharing
(vehicle capacities are 1), with assignments evaluated using
a minimum service time cost function. Figure 4 shows that
predictive positioning is able to reduce customer service times.
When arrival rates are lower than 0.35 ped/min per vehicle,
there is often time between arrivals for vehicles to reposition
to the predictive nodes and service times can be reduced by
up to 20%. As arrival rates increase, however, the benefits of
predictive positioning are reduced as vehicles are continually
allocated to requests. Under high arrival rates, it would be
desirable to add more vehicles to shift to the lower portion
of the curve. However, with a fixed-sized fleet, that option is
not available so ridesharing is used as an alternative.

B. Ridesharing

The performance of the ridesharing algorithm is evaluated by
comparing the single capacity predictive positioning and
unmanaged MOD methods with ridesharing versions
where the maximum vehicle capacity is increased to 3. The
ridesharing algorithm is applied using a minimum service time
cost function. Figure 4 also shows that ridesharing reduces
customer service times. When arrival rates are lower than
0.35 ped/min per vehicle, the ridesharing methods perform
similarly to their single capacity counterparts since current
customers can be serviced before new customers arrive. But
at higher arrival rates, the ridesharing algorithm begins to
utilize the excess vehicle capacity and new customers are
inserted into vehicle schedules before previous customers
have finished their ride. Through the use of a combined
predictive positioning and ridesharing approach, the MOD
system is able achieve a better customer QoS across all arrival
rates, resulting in as much as a 29% reduction in service time
compared to the single capacity unmanaged MOD strategy.

C. Customer Preference

In the previous analysis, the customer preference was
assumed to be focused on service time. However, this
assumption can be wrong and customers may give poor
ratings if the true customer preference lies elsewhere
in other QoS metrics. To evaluate the rating performance
of an MOD system, a simulated rating model is used to
assign a 5-star ratings to customers based on a set of
customer preference weights. The details of the simulated
rating model are provided in Appendix B, which also
specifies how customer preference is encoded using weight
concentration parameters. Six customer preference modes
are analyzed, where the weight concentration parameters are
90% skewed towards either wait time, ride time, service
time, the number of stops while onboard, ride distance, or a
combined weight between service time and ride distance. Five
fleet management strategies are considered. First, a minimum
vehicle distance strategy is considered, where assignments
are not made based on customer ratings but rather based
on the traditional minimum vehicle travel distance metric
which would minimize fuel consumption. Next, three focused
strategies based on ride time, service time, and wait time
are included, where each strategy is given access to the
underlying ground truth ratings function but chooses rating
weights according to its focus. Finally, the presented random
Table I: MOD fleet management performance as measured by average customer rating.

<table>
<thead>
<tr>
<th>Strategy</th>
<th>Customer Preference</th>
<th>Distance</th>
<th>Ride Service</th>
<th># Stops</th>
<th>Distance</th>
<th>Combined</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distance</td>
<td>4.53 4.70 4.22</td>
<td>3.91</td>
<td>4.10</td>
<td>4.17</td>
<td>4.27</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ride Time</td>
<td>4.42 4.97 4.43</td>
<td>4.50</td>
<td>4.97</td>
<td>4.70</td>
<td>4.66</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Service Time</td>
<td>4.67 4.88 4.64</td>
<td>4.30</td>
<td>4.62</td>
<td>4.63</td>
<td>4.62</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wait Time</td>
<td>4.69 4.42 4.09</td>
<td>3.64</td>
<td>3.51</td>
<td>3.82</td>
<td>4.03</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ratings</td>
<td>4.67 4.96 4.63</td>
<td>4.95</td>
<td>4.96</td>
<td>4.60</td>
<td>4.80</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
for each vehicle across multiple machines which bid to a centralized machine for the customer allocations.

Future work will apply these methods to the physical MIT MOD system. Methods for learning and predicting customer arrival rate trends will be studied so that vehicles can be predictively positioned to match real-time demand. Additionally, ratings and service metrics from actual customers will be used to improve and evaluate rating models based on random forest and other machine learning techniques.

ACKNOWLEDGMENT

Research supported by the Ford Motor Company through the Ford-MIT Alliance.

REFERENCES

APPENDIX

A. Computing Customer Ride Metrics

This section provides details on how customer metrics are evaluated using vehicle schedule information. Customer $c$ makes a request at the point in time $t_c$ and is assigned to $v$ currently located at node $n_v$. The pickup $p_c$ and drop off $e_c$ nodes for $c$ occur at respective nodes $s_i$ and $s_j$ in the vehicle schedule $s_v$. The vehicle travels between any adjacent nodes $s_k$ and $s_{k+1}$ in its schedule using route $r(s_k, s_{k+1})$. The travel distance and travel time between the nodes are

\[ d(s_k, s_{k+1}) = \sum_{l \in L(r(s_k, s_{k+1}))} d_l, \]

\[ t(s_k, s_{k+1}) = \sum_{l \in L(r(s_k, s_{k+1}))} d_l/u_l, \]

where $d_l$ and $u_l$ are the length and average travel speed of link $l$, respectively.

The metrics $m_c^{i,j}$ are computed as follows:

\[ t_c^{\text{pickup}} = t(n_v, s_i) + \sum_{k=1}^{i-1} t(s_k, s_{k+1}), \]

\[ t_c^{\text{dropoff}} = t(n_v, s_i) + \sum_{k=1}^{i-1} t(k, s_{k+1}), \]

\[ t_c^{\text{direct}} = t(p_c, e_c), \]

\[ d_c^{\text{direct}} = d(p_c, e_c), \]

\[ t_c^{\text{walk}} = \frac{d(p_c, e_c)}{u_c}, \]

\[ t_c^{\text{ride}} = t_c^{\text{dropoff}} - t_c^{\text{pickup}}, \]

\[ t_c^{\text{wait}} = t_c^{\text{pickup}} - t_c^{\text{request}}, \]

\[ t_c^{\text{service}} = t_c^{\text{wait}} + t_c^{\text{ride}}, \]

\[ t_c^{\text{ratio}} = \frac{t_c^{\text{direct}}}{t_c^{\text{service}}}, \]

\[ t_c^{\text{excess-ride}} = t_c^{\text{ride}} - t_c^{\text{direct}}, \]

\[ N_c^{\text{stops}} = k - j, \]

\[ t_c^{\text{notify}} = t_c^{\text{request}} - t_c^{\text{assigned}}, \]

\[ d_c^{\text{traveled}} = \sum_{k=i}^{i-1} d(s_k, s_{k+1}), \]

\[ t_c^{\text{walk-excess}} = t_c^{\text{service}} - t_c^{\text{walk}}, \]

where $t_c^{\text{pickup}}$ is the point in time $c$ is picked up, $t_c^{\text{dropoff}}$ is the point in time $c$ is dropped off, $t_c^{\text{direct}}$ is the time it would take to drive directly from $p_c$ to $e_c$, $d_c^{\text{direct}}$ is the direct route distance between $p_c$ and $e_c$, $t_c^{\text{walk}}$ is the time it would take to walk from $p_c$ to $e_c$, $t_c^{\text{request}}$ is (9) evaluated with $r(n_i, n_j)$ and $u_l$, as the respective route and speed of the pedestrian instead of a vehicle, and $t_c^{\text{assigned}}$ is the point in time when the ridesharing algorithm assigns the customer to the vehicle. Note, if a customer is rejected ($1^{\text{rej}} = 1$), then many of the metrics do not apply and the customer metrics are set to be $m_c^{\text{rejected}} = \{t_c^{\text{wait}}, t_c^{\text{service}}, t_c^{\text{ratio}}, t_c^{\text{excess-ride}}, N_c^{\text{stops}}, t_c^{\text{traveled}}, t_c^{\text{walk}}, t_c^{\text{walk-excess}}\}$.

B. Simulated Rating Model

This section presents a simulated rating model, which is used as ground truth in simulation to assign 5-star ratings to MOD customers. The values and functional forms are chosen based on an assumed customer preference and are kept hidden from the 5-star rating model.

If a customer is rejected, then they give one of the two lowest ratings based on how long they waited to be notified of their rejection. A rejected customer’s rating is

\[ r_c^{\text{rejected}} = \begin{cases} 2, & \text{if } \frac{t_c^{\text{notify}}}{t_c^{\text{wait}}} \leq 0.1 \\ 1, & \text{otherwise} \end{cases} \]

If a customer is given a ride, then the rating will be a weighted sum of 5 aggregate ratings based on wait time, ride time, service time, number of stops, and ride distance computed as

\[ r_c^{\text{accepted}} = w_1 t_c^{\text{wait}} + w_2 t_c^{\text{ride}} + w_3 t_c^{\text{service}} + w_4 t_c^{\text{stops}} + w_5 t_c^{\text{distance}}, \]

with

\[ t_c^{\text{wait}} = \text{Range}(t_c^{\text{wait}}, 0, 1), \]

\[ t_c^{\text{ride}} = \text{Range}(t_c^{\text{ride}}, 0, 1), \]

\[ t_c^{\text{service}} = \text{Range}(t_c^{\text{service}}, 0, 1), \]

\[ t_c^{\text{stops}} = \text{max}(1, 1 - N_c^{\text{stops}}), \]

\[ t_c^{\text{distance}} = \text{Range}(d_c^{\text{traveled}} - d_c^{\text{direct}}, 0, 0.5), \]

where $\text{Range}(\alpha, \beta, \gamma)$ maps $\alpha$ to the $i$-th interval of 5 exponentially spaced values between $\beta$ and $\gamma$ and assigns the value $6-i$ as the rating. Range values were chosen to reflect a set of possible expected customer satisfaction levels for each metric. For example, setting the $\gamma$ value for $t_c^{\text{distance}}$ to 0.5 reflects that customers would give the lowest rating once their journey distance exceeded the nominal distance by a factor of 0.5. Exponential spacing is used to cause ratings to drop off more quickly as metrics worsen for customers.

The majority of customers follow the same set of weights $\mathbf{w}$, but some customers do not. To accommodate this, the weights are drawn from a Dirichlet distribution such that $\mathbf{w} \sim \text{Dir}(\tilde{w})$, where the concentration parameters $\tilde{w}$ represent the nominal weights for the population.