A data-driven car sharing simulator for inferring latent demand

Technical report

Evan Fields, Carolina Osorio, Tianli Zhou

September 18, 2017

1 Introduction

Forthcoming papers by Tianli Zhou, Carolina Osorio, and Evan Fields [1, 2] make use of a novel car sharing simulator. Those papers contain detailed literature reviews and problem formulations, but brevity and page limits constrain them to include only high level information about our novel simulator. This document, in contrast, gives technical and implementation details about the simulator and is organized as follows: section 2 gives brief motivation of why we developed a new simulator and the main design considerations. Section 3 describes the simulator at an intuitive level, and the remaining sections provide deep details on our mechanisms for generating desired reservations, attempting desired reservations, and calibrating the simulator.

2 Motivating problem

This section briefly outlines the motivating problem which spurred the development of our simulator. For more details, especially on existing literature, please see our forthcoming papers [1, 2]. In our research and this simulator, we consider round-trip station based car sharing. In this car sharing model, users reserve a specific vehicle at a specific station for a pre-specified amount of time, and the user must return the vehicle to that station by the end of the reservation period. A car sharing operator (CSO) generally has data on historical reservations: which cars were used, when the vehicle was used, which user used the vehicle, etc. However, a CSO does not have data on the latent demand—which reservations users *wanted* to make—and critically, the actually made reservations are an imperfect representation of the desired reservations. This happens for many reasons, but the two most important are (i) a user wanting a reservation may find no perfect reservation available, and thus make a less-desired reservation; (ii) a user wanting a reservation may find no perfect or even acceptable reservation available, make no reservation, and thus leave no record in the historical reservation data.

When making operational decisions such as assigning vehicles to stations or planning new stations, a CSO would like to know the amount of latent demand at each station

or potential station. Existing research used one of three strategies for estimating latent demand:

1. Use the observed demand (utilization) as a proxy for latent demand.
2. Use discrete choice models to predict or infer demand for car sharing across a large region, such as an entire city. These methods do not give geographically granular (per-station) estimates of latent demand.
3. Use complicated and computationally expensive meso- or micro-scopic traffic simulators which include a car sharing component. Once properly calibrated—a highly nontrivial task—these simulators can provide demand estimates.

Our industry partner Zipcar (the largest CSO in the United States) wanted a simple tool which could provide actionable insights about latent demand with per-station granularity. The goal of this tool was not to infer latent demand with extreme precision but rather to assist Zipcar in making operational decisions. In particular, the design considerations of the simulator included:

1. *Latent demand trends:* The primary purpose of our simulator was to estimate latent demand and in particular to capture which stations had relatively high or low latent demand.
2. *Simplicity:* The simulator must be simple enough to be explained to Zipcar employees and incorporated into Zipcar's decision making process.
3. *Data-driven:* Zipcar has rich historical data and we agreed the simulator should rely as much as possible on data rather than models of demand and user behavior.
4. *Consistent with expert knowledge:* Zipcar has extensive car sharing experience and knowledge, and it was desireable to embed this knowledge in the simulator, especially if doing so could reduce the difficulty of calibrating the simulator.

3 Key simulator ideas

Our simulator takes as input a study period T, network supply information consisting of the location of each station and the vehicles assigned to each station during period T, and the latent demand d_i of each station i, which is measured in expected reservation-hours wanted per day. The full output of our simulator is station-wise lists of which simulated reservations were made at that station, but the most important outputs (which can easily be computed from the full output) are the station-wise utilizations $\hat{u}_i$, also measured in reservation-hours per day. The central idea of our simulator is that we can infer latent demand through an iterative process of hypothesizing what the latent demand for each station might be, simulating the stations under these hypothesized demands, comparing the simulated utilizations $\hat{u}_i$ to the known historical utilizations u_i, and updating our estimates of d_i. This method of inferring latent demands $\{d_i\}$ is summarized in figure 1.

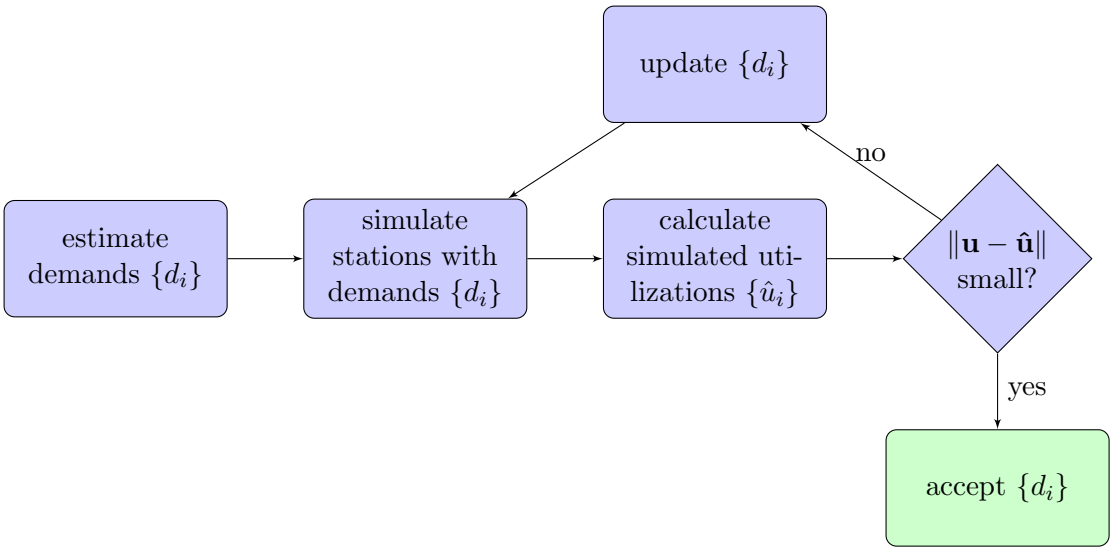

Figure 1: Process for estimating demands $\{d_i\}$

Note that we model each station as having a constant demand for the entire period T, though the way we sample historical reservations accounts for both time-of-day and day-of-week effects. Thus T should not be so long that season-of-year effects vary significantly over T; typically T could be one month. Seasonal effects can be learned by inferring separate demands for each month of the year.
Simulation has three major steps:

1. Generate desired reservations. The latent demands $\{d_i\}$ tell us, in expectation, how many reservations to generate for each station i: enough that the expected reservation-hours wanted per day at station i is d_i. By carefully sampling from the historical data (section 4), we can account for location-dependent patterns in desired reservations, day-of-week effects, and even seasonal effects. A desired reservation is a triple $r = (l(r), t(r), c(r))$, respectively the location the user wants a vehicle, the time interval the user wants a vehicle, and the time the user creates the reservation. Because Zipcar assigns vehicles in a first-come-first-served manner, the creation time indicates which desired reservation can proceed when multiple desired reservations compete for a vehicle.

2. Attempt desired reservations. We simulate users attempting to create each desired reservation ordered by the reservation creation times. Full details are in section 5, but the process can be intuitively summarized: if the desired reservation is available, the user takes that reservation. Otherwise, if an acceptable replacement is available, the user will take a replacement reservation. If no acceptable reservation is available, the user exits without creating a reservation.

3. Compute post-simulation statistics, notably the simulated utilization $\hat{u}_i$ of each station i.

4 Simulator details: generating desired reservations

Desired reservations are generated independently for each station s to be simulated. Here we describe the algorithm for generating desired reservations for station s with demand d_s over period T. In our simulations, the period T is always one or more consecutive calendar days. Let $t_1, \ldots, t_k$ be the calendar days composing T, that is, $\cup_{i=1}^{k} t_i = T$.

1. Let $\mathcal{R}$ be all historical reservations which occurred geographically close to station s. We typically consider reservations from the postal code of station s as this conveniently captures the notion of all reservations which occurred "nearby" station s. When deciding which historical reservations to use to generate demand at station s, there are two competing requirements. On the one hand, we should use as specific data as possible so that the data used best represents the usage patterns of station s; we may even wish to limit to data happening in the same month of the year, etc. On the other hand, we want to use as much data as possible so that when sampling historical reservations, we are sampling from the full breadth of reservations that users wanted and we limit the chance of sampling the same historical reservation multiple times. Empirically, we find these considerations are well balanced by considering all historical reservations from the same postal code as station s, but other definitions are possible.

2. Assign a weight $w(r)$ to each reservation $r \in \mathcal{R}$. In practice, these weights are always pre-computed for all historical reservations, but they could also be computed on the fly as necessary. Section gives details on (pre-)computing the weights $w(r)$.

3. For $i = 1, \ldots, k$, let c_i be the number of reservations in $\mathcal{R}$ which started on the same weekday as t_i. For example, if t_i is a Wednesday, then c_i is the number of reservations in $\mathcal{R}$ which started on a Wednesday.

4. Let $\bar{l} = \sum_{r \in \mathcal{R}} w(r)|t(r)| / \sum_{r \in \mathcal{R}} w(r)$ be the weighted average reservation length in $\mathcal{R}$. Choose n, the number of reservations to generate for station s, by $n \sim \text{Pois}\left(\frac{k \cdot d_s}{\bar{l}}\right)$.

5. Initialize A, the list of generated reservations, to be empty. Repeat the following n times (each iteration generates a single desired reservation):

   (a) Choose a day t_i from $\{t_1, \ldots, t_k\}$ where each day t_i is sampled with probability proportional to c_i. The generated reservation will start on day t_i.

   (b) Choose a historical reservation r from $\{r \in \mathcal{R} : r \text{ starts on the same weekday as } t_i\}$.

   (c) Let r' be a reservation with location $l(r') = s$, start time of day (e.g. 3:30pm) the same as reservation r's start time of day, length $|t(r')| = |t(r)|$, start day t_i, and start time $c(r')$ such that the interval between $c(r')$ and $t(r')$ matches the interval between $c(r)$ and $t(r)$. For example, if historical reservation r was created 3 hours before the reservation time, r' should also have creation

time 3 hours before the start of its reservation time. Intuitively, we have made a "copy" of r at location s and day t_i. Add r' to A.

6. Return A, which now contains n desired reservations.

Observe that by limiting the historical reservations to those reservations near station s, we have captured geographic trends in demand. And by first sampling a day in T and then considering historical reservations from the same weekday, we capture day of week effects.

4.1 Choosing weights w

As mentioned in section 2, the historical reservation data represents a truncated sample of users' first-choice reservation preferences. If we sample uniformly at random from the historical data to generated desired reservations in simulation, then the simulated successful reservations will actually represent a *twice* truncated sample of users' first choice preferences.

An example is illustrative: suppose that users who want long reservations often have difficulty finding a vehicle matching their needs, primarily because a single short reservation can block a user from reserving an otherwise-available vehicle for a long reservation. Further suppose that users who want short reservations can usually find a vehicle; their first choice location-time preference may not be available, but if a user just wants a car for an hour, there's likely a vehicle nearby which is available. Then the historical reservation data will include most of the short reservations that users wanted but fewer of the long reservations. When sampling reservations from the historical data to serve as simulated desired reservations, we should sample the observed long reservations with slightly higher probability than the observed short reservations.

The method of sampling from the historical data according to weights $w(\cdot)$ on each reservation should be seen as a detruncation method. For full details on this detruncation method, including a literature review and validation experiments, please see [1].

From our conversations with car sharing stakeholders, we believe that the length of a desired reservation is one of the most important factor determining whether that reservation can be realized. Further, we observe that if we set all weights $w(r) = 1$ (i.e. sample uniformly from historical data) that the average length $|\bar{t}|_{\text{sim}}$ of simulated successful reservations is about 15 minutes shorter than the average length $|\bar{t}|_{\text{data}}$ of historical reservations. Therefore, we choose weights $w(r) = 1 + \alpha \cdot |t(r)|$ where $|t(r)|$ is the length of reservation r. To choose the parameter α, we solve the following one-dimensional simulation based optimization problem:

$$\begin{aligned} \underset{\alpha}{\text{minimize}} \quad & \mathbb{E}\left[|\bar{t}|_{\text{data}} - |\bar{t}|^w_{\text{sim}}\right]^2 \\ \text{subject to} \quad & w(r_i) = 1 + \alpha \cdot l(r_i) \quad \forall r_i \in R \\ & w(r_i) \geq 0 \quad \forall r_i \in R \end{aligned} \tag{1}$$

where $|\bar{t}|^w_{\text{sim}}$ is the average simulated reservation length under weights w. Because the problem is one-dimensional and $\mathbb{E}\left[|\bar{t}|^w_{\text{sim}}\right]$ is monotonically increasing in α, we can solve

this problem with a simple line search or binary search and find that a value of $\alpha = .0016$ works well. This may seem like a tiny value, but it means that a 24-hour reservation is about 4% more likely to be sampled than a 1-hour reservation, and under this weighting scheme the average simulated reservation length matches the average historical reservation length.

5 Simulator details: attempting a reservation

This section describes the algorithm we use to capture the intuitive notion that when a customer wants a reservation, they take their first-choice reservation if it is available, otherwise they search for an acceptable substitute, and if no acceptable substitute can be found, they exit the system.

For a reservation $r = (l(r), t(r), c(r))$, let $\mathcal{S}(r) = \{r' : |t(r)| = |t(r')| \text{ and } c(r) = c(r')\}$. That is, $\mathcal{S}(r)$ is the set of all possible substitute reservations for r which we define as reservations with the same length and creation time as r; we allow substitute reservations for r to differ from r only in location and start time. We define a distance $\delta : \mathcal{S}(r) \times \mathcal{S}(r) \to \mathbb{Z}_{\geq 0}$ on the space of substitute reservations, and let $p \in (0, 1)$ be a user choice parameter representing users' willingness to accept substitute reservations. Note that we keep p constant across all users and desired reservations. See section 6 for how p and δ are chosen. Then the algorithm for a user attempting a desired reservation r is

1. If one or more vehicles are available at location $l(r)$ for time period $t(r)$, the user selects one of the vehicles uniformly at random and reserves that vehicle for period $t(r)$. The algorithm then exits.

2. Otherwise, let $\varepsilon = 0$ and $x_\varepsilon = -1$. While $x_\varepsilon < p$:

   (a) Let $\mathcal{S}_\varepsilon = \{r' \in \mathcal{S}(r) : r' \text{ is available and starts no more than 30 minutes before } c(r)\}$. $\mathcal{S}_\varepsilon$ is the set of available substitutes for first-choice reservation r at distance ε from r and that can be created at creation time $c(r)$. Note that we allow reservations which begin slightly in the past; e.g. at 4:05pm a reservation starting at 4pm can be created, but a reservation starting at 2pm could not be created.

   (b) If $\mathcal{S}_\varepsilon$ is nonempty, choose r' uniformly at random from $\mathcal{S}_\varepsilon$. Because r' is available, at least one vehicle at location $l(r')$ is available over period $t(r')$. The user chooses one of these vehicles uniformly at random and reserves this vehicle for period $t(r')$, and the algorithm exits.

   (c) Otherwise, the user decides whether to consider more dissimilar substitute reservations. Let $\varepsilon = \varepsilon + 1$ and $x_\varepsilon \sim \text{Unif}(0, 1)$.

3. The user was unable to find an acceptable substitute reservation, so the algorithm exits without any reservation being made.

This process is summarized visually in figure 2.

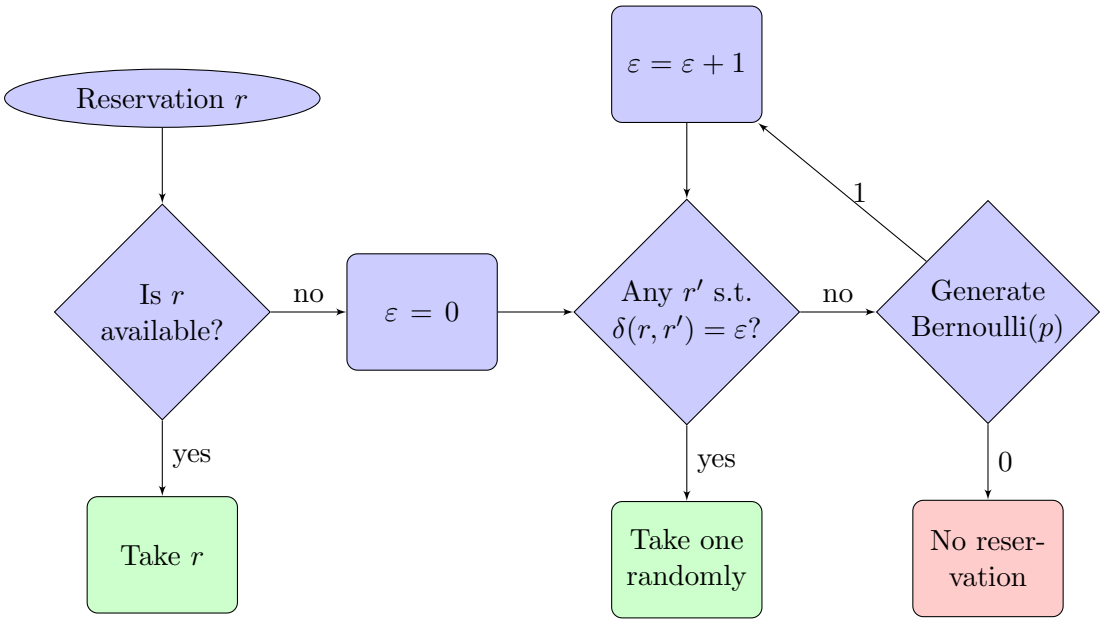

Figure 2: Attempting a reservation r

6 Simulator details: calibrating user choice parameters

As input, our simulator requires station-wise demands $\{d_i\}$ as well as two parameters p and δ which describe user behavior when choosing a possible substitute reservation when the first-choice reservation is unavailable. In theory both the demands and user choice parameters could be jointly calibrated using simulation based optimization, by matching the simulated utilizations $\{\hat{u}_i\}$ to the historically observed utilizations $\{u_i\}$. However, there are at least three problems with this approach:

1. Our industry partner Zipcar has extensive experience with and data on user behavior, including from observational studies and user surveys. Calibrating the user choice parameters solely by simulation based optimization forgoes this important knowledge. The calibration formulation could include a regularization term which nudges the calibrated user choice parameters towards the prior knowledge, but any reasonable weight on the regularization term would be so large that the calibrated user choice parameters could not strongly differ from the prior knowledge.

2. The combination of demands $\{d_i\}$ and user choice parameters which minimize the calibration error may be non-unique. In particular, to reproduce a given historical utilization u in simulation, a high demand d is needed if customers have rigid preferences and tend to leave the system if their first-choice reservation is unavailable, and conversely a lower demand d will suffice if customers are highly flexible. This dependency between demands and choice parameters complicates both solving the calibration problem and interpreting the results.

3. Jointly calibrating demands and choice parameters introduces extra dependency between demand parameters. In general if the choice parameters p and δ are fixed, the demands for two stations i and j which are geographically far from one another will have negligible impact on one another. That is, knowing demand d_i neither constrains nor provides information about demand d_j. However, if the choice parameters are not fixed then there is direct dependency between each demand and the choice parameters and thus "second-order" dependency between different demand parameters. This prevents any kind of geographic partitioning into subproblems. (See section 7.)

In light of these reasons, we use stakeholder expert knowledge to choose the user choice parameters p and δ. This is especially appropriate in that our simulator is designed primarily to learn *trends* in latent demand. So, for example, if we choose too high a value of p (customers are modeled as too flexible), the inferred demands d_i will tend to be too low—but this will be true for every station, and we'll still capture the patterns of which stations are net sources and attractors of spillover demand.

p and δ are chosen so that the reservation choice behavior in the simulator matches the choice behavior of real customers. From conversations with Zipcar personnel and from Zipcar user surveys we know that customers dislike walking more than half a kilometer to reach a vehicle, but we also know that distances of less than 100 meters are seen as no problem. So p and δ should be chosen so that few simulated customers would accept a reservation at a station one kilometer from their desired station and most would accept a reservation 50 meters from their desired station.

The reservation dissimilarity function δ must include both spatial and temporal distance between two reservations. We combine these in the simplest way possible: by adding the spatial and temporal distances between two reservations; $\delta = \delta_{\text{spatial}} + \delta_{\text{temporal}}$. Let us first consider reservations with the same location but different times. In our data, all reservations start on an hour or half-hour, leading to the natural definition that δ_{temporal} between two reservations is the number of half-hour blocks separating their start times.

From this we can estimate a reasonable spillover probability p. Consider a customer choosing between reservations at that customer's desired locations but varying times. Typical customers will accept a reservation around one and a half hours offset from their desired reservation, suggesting a spillover probability $p = \frac{3}{4}$.

We can similarly choose δ_{spatial}. Given that customers have negligible aversion to very short spatial distance and high aversion to long distances, δ_{spatial} should be convex. We choose a simple quadratic, $\delta_{\text{spatial}} = d^2/100,000$, rounded to the nearest integer, where d is the crow-flies distance in meters between two stations. This captures the notion that distances on the order of 100 meters are generally insignificant but distances above half a kilometer are untenable for most customers. The reservation attempt algorithm in section 5 requires integer dissimilarities δ between reservations, so we round δ_{spatial} to the nearest integer; by construction, δ_{temporal} is already integer valued.

7 Simulator details: inferring demands d_i

Our simulator, which is implemented in Julia, is fairly efficient: on average, around 9000 station-days can be simulated per second. In a large study region (we use the Boston, United States metropolitan region as our case study), there may be hundreds or thousands of stations. Thus simulating the entire study region for a period of a month takes around two seconds. When inferring the station-wise demands $\{d_i\}$, however, the relative efficiency of the simulator is offset by the high number of demands to learn. In Boston there are approximately 500 different stations, so using a basic finite difference approach (analytic derivatives are not available) to estimating the gradient of a network-wide objective function with respect to station demands would take on the order of 1000 seconds, which is quite problematic. Instead of relying on complicated cutting-edge simulation based optimization techniques, we take advantage of problem structure which encourages a natural geographic decomposition into smaller problems.

Intuitively, two stations have little effect on each other if they are far apart. As an extreme, consider two stations 10km from each other on different islands. The demand at one station will have no effect on the other; no users wanting a reservation at one station will instead take a reservation at the other station separated by 10km and a body of water. The demand for each of these stations can be learned independently: rather than solving one calibration problem for two stations, we can solve two problems for one station each. The same intuition applies in more realistic cases. If two stations are far apart, they have little impact on each other, and what impact they do have is carried through intermediate stations. As an example, suppose stations a, b, c are arrayed in a line such that the distance from a to b and from b to c are both one kilometer. A small amount of demand can spill from a to b and b to c, but no almost no demand will spill directly from a to c. So when simulating station a, it's sufficient to simulate just station b to catch and generate spillover demand; station c is not necessary.

Let T be the study period (for which we have historical data), typically 2-8 weeks long. Let $\mathcal{L}$ be the set of stations for which we want to infer demands $\{d_i\}$ for period T. Let $L_1, \ldots, L_n$ partition $\mathcal{L}$, and for each L_i, let $\bar{L}_i$ be the union of L_i and the k stations closest to L_i which are not in L_i. In practice we typically take $k = 10$. If the partition of $\mathcal{L}$ into L_i is well chosen, then the k stations in $\bar{L}_i \setminus L_i$ serve as "boundary stations" which manage the interaction between L_i and the rest of the network. We solve a simulation-based calibration problem to learn the demands for each $\bar{L}_i$ independently, and from each we record the inferred demands for the non-boundary stations in L_i. Note that in general a given station may be a boundary station in multiple $\bar{L}_i$, but because the $\{L_i\}$ partition $\mathcal{L}$, each station will be in exactly one L_i and thus we will learn a single demand d_s for each station s. To infer demands for stations s within a single $\bar{L}$, we use the following coordinate descent inspired algorithm:

1. For each $s \in \bar{L}$, initialize d_s to an initial guess as follows:
   (a) Let $\eta > 0$ be an integer; we typically use 10. Let $d_{\min} = 0$, $d_{\max} = 100$, $d_{\text{step}} = 1$.

(b) Let $\mathbf{x} = d_{\min}, d_{\min} + d_{\text{step}}, d_{\min} + 2d_{\text{step}}, \ldots, d_{\max}$.

(c) For each $x_i \in \mathbf{x}$, let $y_i = \frac{1}{\eta} \sum_{j=1}^{\eta} \hat{u}_s^j$ where $\hat{u}_s^j$ is the simulated utilization of station s on the j-th simulation run, simulating only station s.

(d) Fit a second order local polynomial regression to data $(\mathbf{x}, \mathbf{y})$. We use the Julia package `Loess.jl`. Denote by f this regression, i.e. $f(x_i) \approx y_i$.

(e) If $f(d_{\max}) \geq u_s$, the actual historical utilization of station s, then continue. Otherwise double $d_{\max}$ and d_{step} and go to step (b).

(f) Let $d_s = \min\{x \in d_{\min}, d_{\min} + 0.01, d_{\min} + 0.02, \ldots, d_{\max} : f(x) \geq u\}$, that is, the smallest (to a granularity of .01 reservation-hours per day) demand which we simulate as producing a utilization at least as high as the historically observed utilization.

The intuition behind this initialization procedure is to capture the correct demand for each station if spillover between stations did not occur. This should give an order of magnitude correct estimate for each station, even in the presence of spillover.

2. We then adjust for spillover. Let $m_{\max} > 0$ be the maximum number of iterations; we typically use 5. Let $\varepsilon > 0$ be a convergence tolerance; we use $.1\sqrt{|\bar{L}|}$. For $m = 1, \ldots, m_{\max}$ do the following:

(a) Let σ be chosen uniformly at random from all permutations of $1, 2, \ldots, |\bar{L}|$.

(b) Let $\eta = \lceil m/2 \rceil$.

(c) For $s \in \sigma(1), \ldots, \sigma(|\bar{L}|)$:

i. Let $\mathbf{x} = .5d_s$, $.65d_s$, $.8d_s$, $\ldots$, $2d_s$. For each $x_i \in \mathbf{x}$, let

$$y_i = \| \frac{1}{\eta} \sum_{j=1}^{\eta} \hat{\mathbf{u}}^j - \mathbf{u} \|_2^2 \tag{2}$$

where $\hat{\mathbf{u}}^j$ is the vector of simulated utilizations of all stations in $\bar{L}$ on the j-th simulation run, and similarly $\mathbf{u}$ is the vector of historical utilizations of each station in $\bar{L}$ over period T.

ii. Fit a second order local polynomial regression to data $(\mathbf{x}, \mathbf{y})$. Call this regression f.

iii. Update d_s to

$$\text{argmin}\{f(.5d_s), f(.5d_s + .01), \ldots, f(2ds)\}$$

Informally speaking, the second step repeatedly performs single coordinate line searches. The cumulative effect of these single-coordinate adjustments is to adjust for the effects of demand spilling from one station to another. Note that in the calibration objective (equation 2), we match average simulated utilization over period T to historical utilization over period T. Under any given set of demands $\{d_i\}$, the simulated utilizations $\hat{\mathbf{u}}$

have high variance, and thus we average multiple simulation runs. This does mean that if the historical utilizations $\mathbf{u}$ contain some atypical values due to inherent randomness in the car sharing system, the demands $\{d_i\}$ will be chosen so as to in expectation reproduce these atypical values. This effect could be reduced via some regularization or prior belief on the station-wise demands $\{d_i\}$, but we do not use such techniques in this work.

A Notation

Symbol	Meaning
d_i	latent demand at station i, measured in reservation-hours per day
u_i	historical utilization at station i, measured in reservation-hours per day
$\hat{u}_i$	simulated utilization at station i, measured in reservation-hours per day
$l(r)$	location of reservation r
$t(r)$	time interval of reservation r
$c(r)$	creation time of reservation r
$\mathcal{S}(r)$	reservations substitutable for desired reservation r

References

[1] Evan Fields, Tianli Zhou, and Carolina Osorio. A data-driven method for reconstructing a truncated distribution with an application to inferring car sharing demand. 2017, forthcoming.

[2] Tianli Zhou, Carolina Osorio, and Evan Fields. A discrete simulation-based optimization algorithm for two-way car-sharing network design. 2017, forthcoming.