Using Sample Average Approximation to Fulfill On-Demand Requests with Increased Driver Autonomy

Hannah Horner, Jennifer Pazour, John E. Mitchell
Rensselaer Polytechnic Institute, horneh@rpi.edu, pazouj@rpi.edu, mitchj@rpi.edu

1. Problem Statement and Motivation

Today's society is facing an ever-increasing demand for prompt local transportation of passengers and products. An emerging strategy for addressing these needs is the use of Ridesharing and Crowdsourced Delivery. These are on-demand peer-to-peer logistics platforms, where a fleet of non-professional ad hoc drivers are compensated to locally transport a customer or a product. A plethora of companies, such as Uber and Grubhub, apply such sharing economy-based business models.

As recently reviewed in Mourad, Puchinger, and Chu (2019), a growing body of literature introduces optimization models to match available drivers to requests to optimize some performance measure(s), typically corresponding to the platform's utility. To our knowledge, all of these models, except the model presented in Mofidi and Pazour (2019), offer a single request to each driver. While the model in Mofidi and Pazour (2019) does offer choice to the drivers in the form of a personalized menu of requests to choose from, a deterministic optimization model is created and used to illustrate that driver choice is beneficial to a platform when the platform is uncertain about drivers' preferences. However, the deterministic model is not able to determine optimal menus for drivers, as it is shown that an optimal solution set contains a menu set with at most one option per driver. Though some models restrict inconveniences to the driver, such as extra driving time or the number of pickup/dropoff locations, to our knowledge all models with the exception of those from Mofidi and Pazour (2019) and Soto Setzke et al. (2017) do not allow the driver to reject their matched request. Our contribution is a single level stochastic integer program model able to create personalized menus of request recommendations for drivers, using a stochastic driver behavior model that reflects the platform's uncertainty in driver behavior.

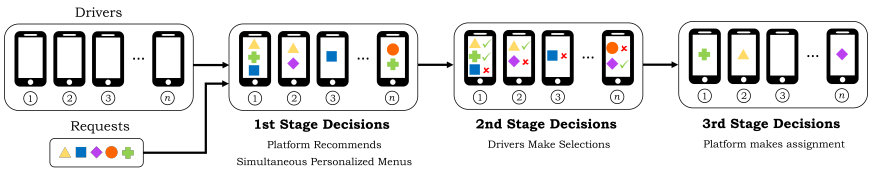


Figure 1 **The three decision stages of our model: the first stage decision, second stage stochastic event, and third stage recourse decision.**

There is clear incentive to design and optimize methods to match drivers to requests, as fulfilling requests efficiently is integral to the success of the platform. Because uncertainty in driver behaviour exists, offering multiple options to each driver has the potential to increase the probability that a driver accepts at least one request. Having more assignment options for the platform can allow for more platform-efficient assignments, while offering multiple requests provides drivers with a higher level of autonomy. As a platform will have at least some uncertainty in driver behavior, a stochastic model can capture the different possible driver feedback outcomes. However, incorporating these elements into an optimization model is not without its difficulties, as a multistage stochastic program is computationally expensive.

Our method is a form of Stackelberg game, with an overview of the process depicted in Figure 1. The first stage leader decisions are platform decisions: it must determine the personalized menu to send to each driver that will maximize the platform's expected utility. The second stage is then comprised of the drivers' feedback on the menus they received in Stage 1. In this stage, each driver will select one or more requests from their menu to fulfill or opt to not fulfill any of the requests. The third and final stage is the platform's processing of this feedback and determining the optimal driver-request assignment that is feasible given driver responses from Stage 2. Then the list of available drivers and requests is updated and the platform reoptimizes the menu set to send out in the next time period. Our model focuses on a single time period with one simultaneous round of recommendation menus, driver feedback, and assignments.

2. Approach

To capture driver behaviour that is computationally able to integrate with an optimization model, we assume a driver's decision to accept or reject each request is independent of any other requests in their menu; their decision depends only on their personal preference of that request. More specifically, for a given driver, the preference of a given request is quantified as the net personal utility to the driver: the utility the driver receives for fulfilling the request minus the 'no-choice' utility the driver receives from electing to not fulfill any request. Each driver's utility is assumed to not be influenced by the utilities of other drivers. The drivers are assumed to rationally make selections that maximize their utility. The platform is unable to perfectly estimate the drivers'

utilities for requests and this is captured by modeling the utilities as random variables with known distributions.

Given a full set of net utility value realizations for all driver and request pairs, a preference ranking for each driver can determine which request(s), if any, the drivers will accept for any given menu. We consider different realizations to be different 'scenarios,' allowing us to explicitly model the stochastic problem as a bilevel integer program. If there are m requests and n drivers, there can be as many as $((m+1)!)^n$ unique scenarios. This is far too many to model and solve in a reasonable amount of time for a problem with more than a few drivers and requests, so we employ the Sample Average Approximation to provide the optimal menu sets over an approximated model of driver behaviors. However, even using this approximation method, computational struggles still exist. To develop a computationally viable model, we capitalize on a special case that occurs when drivers are allowed in Stage 2 to accept as many requests from their menu as they like. Our initial formulation allows the platform to limit the number of requests a driver may select, for example, having them elect only their favorite request. Relaxing this constraint to give drivers unlimited selection changes the way we can model their behavior: instead of using a request preference ranking, the likelihood that a driver will accept a request is independent of other requests and can be modeled using Bernoulli random variables. Redefining the behavior model, we are able to reformulate our bilevel formulation to an equivalent single level formulation with only 2^{mn} or fewer scenarios.

3. Results and Conclusions

Input parameters such as platform utility and the probability that drivers will accept each request are calculated using data from a realistic transportation network representing Chicago and the fare rates used by Lyft in Chicago. The driver will specify their original trip, planned for their own purposes, and using the transportation network we estimate the extra driving time for the driver if they do a platform pickup and dropoff for a request during their trip. The fare collected by the platform and by the driver is also calculated and this information combined with the extra driving time is used to calculate the full set of input parameters. To analyze the performance of a menu produced by our model, we randomly generate a set of test scenarios and calculate the driver feedback and optimal recourse assignment for each scenario. The final objective value for each scenario is then averaged to form a performance estimate for the menu. In exploratory testing we use a solver time limit of 500 seconds and a maximum optimal solution gap of 1%.

An open question is to determine an appropriate test scenario size, as an accurate performance measurement is important. Increasing sample size will increase accuracy, but evaluating a scenario involves solving an integer program, so the time required increases linearly with the number of the test scenarios being evaluated. We conduct an analysis on smaller problems with up to 10 drivers

and 10 requests where we can compute the true error of our performance estimates, as well as for larger problems with up to 40 drivers and 40 requests. For 8 out of 9 smaller problems, the estimate produced using 5,000 scenarios (out of a total of ∼16,000-200,000) is always within 1.9% of the true performance value and requires an average of 168.3 seconds to compute. For all of the larger problems tested, the estimate obtained using 5,000 samples requires an average of 331.6 seconds and is always within 2.4% of a performance estimate calculated using 100,000 scenarios. This means for both large and small problems, we can save a significant amount of time and achieve similar performance estimates using only a small number of test scenarios.

We next determine how many scenarios should be generated for the optimization model, i.e. the size of the training scenario set. We calculate the true optimal solution for small problems with up to 6 drivers and 6 requests, and find that with a sample size of 100 (out of ∼1,000-8,000 scenarios), the performance of the menu produced is always within 22.3% of the optimal menu's performance. A similar analysis performed on a set of larger problems illustrates that menus produced using 100 scenarios perform within 9.3% of the best-found menu using 1,000 scenarios, which is lower than the maximum error of menus produced in 500 seconds using larger sample sizes. On average, making a menu for 40 drivers and 40 requests using 100 scenarios requires only 14.3 seconds, meaning we are able to generate driver menus for decently-sized problems in a reasonable amount of time.

Now armed with a method to create personalized menus, we will explore the properties of the produced menus and the effects of different factors such as the maximum menu size, driver compensation, etc. This analysis will provide insight into questions such as 'How much driver willingness to participate is needed to have a functioning platform?' 'What wage level for drivers has the highest utility to the platform?' 'What is the ideal menu size?' Answers to questions like these will allow platforms to design ridesharing and crowdsourced delivery systems with increased driver autonomy.

Acknowledgments

This work was partially funded by the National Science Foundation CAREER award 1751801 and the Johnson & Johnson WiSTEM2D fellowship.

References

Mofidi S, Pazour J, 2019 *When is it beneficial to provide freelance suppliers with choice? a hierarchical approach for peer-to-peer logistics platforms. Transportation Research Part B Methodology* .

Mourad A, Puchinger J, Chu C, 2019 *A survey of models and algorithms for optimizing shared mobility. Transportation Research Part B: Methodological* .

Soto Setzke D, Pflügler C, Schreieck M, Fröhlich S, Wiesche M, Krcmar H, 2017 *Matching drivers and transportation requests in crowdsourced delivery systems.* Twenty-third Americas Conference on Information Systems.